МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Вебзастосунок для генерації 2d sketch на основі генеративного штучного інтелекту»

Виконавець: Ігор МАКАРЧУК

Керівник: к. пед. н., доцент Юрій СІНЬКО

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Макарчука Ігоря Руслановича
(ПІБ випускника)

1. Тема роботи: «Вебзастосунок для генерації 2d sketch на основі генеративного штучного інтелекту» затверджена наказом ректора № 1976/ст від 29.09.2023р.
2. Термін виконання роботи: з 02 жовтня 2023 року по 31 грудня 2023 року.
3. Вихідні дані до роботи: вебзастосунок на мові програмування JS для демонстрації мікросервісної архітектури.
4. Зміст пояснювальної записки: 1. Теоретичні основи генеративного штучного інтелекту. 2. проектування застосунку. 3. Розробка додатку.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Користувацький інтерфейс додатку. 2. Компонент "Опис Відео". 3. Компонент "Редагування", 4. Майбутній дизайн додатку. 5. Atomic Design Methodology. 6. Схема роботи React. 7. Користувацький інтерфейс для покращення моделі.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області. Написання 1 розділу, аналіз основ генеративного штучного інтелекту.	02.10.2023-18.10.2023	
2.	Проектування застосунку до вимог. Написання 2 розділу, проектування застосунку.	19.10.2023-31.10.2023	
3.	Написання 3 розділу, розробка застосунку.	01.11.2023-15.11.2023	
4.	Загальне редагування та друк пояснювальної записки.	16.11.2023-20.11.2023	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	21.11.2022-16.12.2022	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	16.12.2023-22.12.2023	

7. Дата видачі завдання 2.10.2023 р.

Керівник кваліфікаційної роботи _____ Юрій СІНЬКО
(підпис керівника)

Завдання прийняв до виконання _____ Ігор МАКАРЧУК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Вебзастосунок для генерації 2d sketch на основі генеративного штучного інтелекту» містить: 90 сторінок, 25 рисунків, 21 інформаційне джерело.

Об'єкт дослідження – процес генерації 2d ескізів з використанням генеративного штучного інтелекту при проектуванні його компонентів

Предмет дослідження – методи та засоби генеративного штучного інтелекту для генерації зображень.

Мета кваліфікаційної роботи – дослідження напрямків і технологій ГШІ для генерації зображень та розробка веб-додатку, який використовує ГШІ для створення 2D ескізів на основі текстових запитів, спрощуючи процес пре-продакшену відеоконтенту та генерації ідей.

Методи дослідження – модель Stable Diffusion, Chat GPT API, Мова програмування Typescript, інтегроване середовище розробки VSCode

Результати кваліфікаційної роботи рекомендується використовувати для демонстрації можливостей створення 2d sketch та в подальшій інтеграції у рішення компаній та підприємств.

ЗАСТОСУНОК, ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, STABLE DIFFUSION, CHAT GPT API, TYPESCRIPT, WEB

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

IDE (*Integrated Development Environment*)	−	Інтегроване середовище розробки
NLP (Natural language processing)	−	Обробка природної мови
SD (Stable Diffusion)	–	Назва моделі
VAE (Variational autoencoder)	–	Варіаційний кодувальник
GAN (Generative Adversarial Network)	–	Генеративна змагальна мережа
DOM (Document Object Model)	–	Об'єктна модель документа
ШІ (Штучний інтелект)	–	Термін
ГШІ (Генеративний штучний інтелект)	–	Термін
CV (Computer Vision)	–	Комп'ютерний зір

ВСТУП

У сучасному світі швидкого розвитку цифрових технологій, особливе місце займає галузь штучного інтелекту, зокрема в області генеративних моделей. Цей напрямок ШІ відкриває безмежні можливості для створення візуального контенту, викликаючи значний інтерес у сфері відео, реклами, освіти та розваг. Враховуючи це, актуальність розробки веб-додатків, які можуть автоматизувати та спростити процес створення візуального контенту, є незаперечною.

Дана робота присвячена розробці та дослідженню веб-додатку, що дозволяє генерувати 2D-зображення на основі текстових описів за допомогою генеративного штучного інтелекту.

Метою дослідження є технологія ГШІ для генерації зображень та розробка веб-додатку, який використовує ГШІ для створення 2D ескізів на основі текстових запитів:

1. Здійснити огляд та аналіз моделей генеративного штучного інтелекту, визначити її переваги та недоліки.
2. Визначитися з користувацьким інтерфейсом та функціоналом додатку та його організацією.
3. Обрати інструменти та технології для розробки
4. Розробити повнофункціональний вебзастосунок для генераціх 2d ескізів на базі генеративного штучного інтелекту.

Предметом досліджень є методи та засоби генеративного штучного інтелекту для генерації зображень.

Об'єктом дослідження є процес генерації 2d ескізів з використанням генеративного штучного інтелект.

Актуальність теми полягає у зростаючій потребі цифровізації та автоматизації в процесі створення візуального контенту, особливо відео. Наприклад на ринку існують наступні проблеми які вирішує данна робота.

1. Створення комплексної концепції та визначення візуального стилю - дуже складний і трудомісткий процес.

2. Між клієнтом та креативним менеджером існує постійний розрив у спілкуванні через брак візуального розуміння

3. Молодший ілюстратор з усіх сил намагається створити ілюстрацію, а результати його роботи не влаштовують клієнта, що призводить до збільшення витрат для бізнесу та зриву дедлайнів

Аудиторією такого продукту можуть бути наступні категорії.

– креативні директори з досвідом роботи в ілюстраціях до 10 років і майже нульовими навичками малювання

– моушн-дизайнери, які хочуть отримати готовий до затвердження розкадрування для клієнта Ілюстратори з досвідом роботи з ілюстраціями від нуля і майже нульовими навичками малювання:

– відеоагенції, які хочуть отримати готову розкадровку для клієнта;

– ілюстратори з досвідом роботи з ілюстраціями та майже нульовими навичками малювання;

– кінцеві клієнти з досвідом роботи з ілюстраціями до 10 років і майже нульовими навичками малювання;

– маркетологи, які хочуть отримати готову розкадровку для клієнта;

Наукова новизна роботи полягає в можливості генерації скетчів які відповідають сучасним вимогам ринку без прямого втручання людини, надаючи відеоагенціям та їх клієнтам ефективні та швидкі рішення.

Важливість дослідження полягає у вивченні та застосуванні передових технологій у сфері штучного інтелекту, зокрема використанні моделі Stable Diffusion, для автоматизації творчого процесу. Такий підхід дозволяє не тільки спростити розробку візуального контенту, але й відкриває нові горизонти для експериментів у дизайні, маркетингу та інших креативних індустріях.

У рамках цієї роботи будуть розглянуті теоретичні основи генеративного штучного інтелекту, аналіз існуючих рішень у сфері створення візуального контенту, а також детально описано процес розробки та функціональні можливості веб-додатку "Ideations".

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

1.1. Історія розвитку генеративного ШІ

Генеративний штучний інтелект (ГШІ) - це галузь штучного інтелекту, яка займається розробкою алгоритмів і методів для створення нових даних, таких як текст, зображення, музика та відео. ГШІ має широке застосування в різних сферах, таких як створення контенту, машинний переклад, розпізнавання образів та автоматизоване проектування.

Історія розвитку ГШІ можна умовно розділити на декілька етапів:

Ранній етап розвитку (1950-1980-ті роки)

Перші дослідження в галузі ГШІ були проведені в 1950-х роках. На цьому етапі основними методами ГШІ були статистичні методи, такі як стохастичне моделювання та генеративні моделі.

Стохастичне моделювання використовувалося для генерації випадкових даних, які відповідали деяким заданим характеристикам. Наприклад, можна було використовувати стохастичне моделювання для генерації випадкових текстових даних, які відповідали заданій кількості слів, синтаксису та стилю.

Генеративні моделі використовувалися для генерації нових даних на основі наявних даних. Наприклад, можна було використовувати генеративну модель для генерації нових картин на основі наявної бази даних картин.

Етап машинного навчання (1980-ті - 2000-ні роки).

У 1980-х роках в ГШІ почали активно застосовуватися методи машинного навчання, такі як нейронні мережі та підтримка векторних машин.

Нейронні мережі - це тип машинного навчання, який моделює людський мозок. Нейронні мережі можуть використовуватися для генерування різних типів контенту, таких як текст, зображення та музика.

Підтримка векторних машин - це метод машинного навчання, який використовується для класифікації та регресії. Підтримка векторних машин може використовуватися для генерації контенту, який відповідає певним критеріям.

Етап глибокого навчання (2000-ні - сьогодення)

У 2000-х роках розвиток ГШІ отримав значний поштовх завдяки появі технології глибокого навчання. Глибоке навчання - це підгалузь машинного навчання, яка використовує нейронні мережі з великою кількістю шарів. Глибоке навчання дозволяє розробити більш складні і ефективні генеративні моделі, які можуть створювати високоякісний контент.

Одним з найважливіших досягнень в галузі глибокого навчання в контексті ГШІ було створення моделі Generative Adversarial Network (GAN) в 2014 році. GAN - це тип генеративної моделі, яка використовує два типи нейронних мереж: генератор і дискримінатор. Генератор створює новий контент, а дискримінатор намагається відрізнити створений контент від справжнього.

GAN дозволили значно підвищити якість генерованого контенту. Наприклад, GAN можна використовувати для генерації реалістичних зображень людей, тварин, об'єктів та сцен.

Після 2014 року розвиток ГШІ продовжувався. Були розроблені нові генеративні моделі, які дозволяють створювати ще більш реалістичний і якісний контент.

Ось деякі з найважливіших досягнень у галузі ГШІ після 2014 року:

У 2017 році компанія OpenAI представила модель генеративної трансформаторної мережі (GPT-2), яка може генерувати текст, який важко відрізнити від написаного людиною.

У 2020 році компанія Nvidia представила модель генеративної трансформаторної мережі (StyleGAN2), яка може генерувати реалістичні зображення людей, тварин та інших об'єктів.

У 2022 році компанія Google представила модель генеративної трансформаторної мережі (LaMDA), яка може генерувати текст, який відповідає заданим запитам і вимогам.

Ці досягнення свідчать про те, що ГШІ має потенціал революціонізувати різні сфери життя. ГШІ можна використовувати для створення більш якісного контенту, вирішення практичних завдань і створення нового мистецтва.

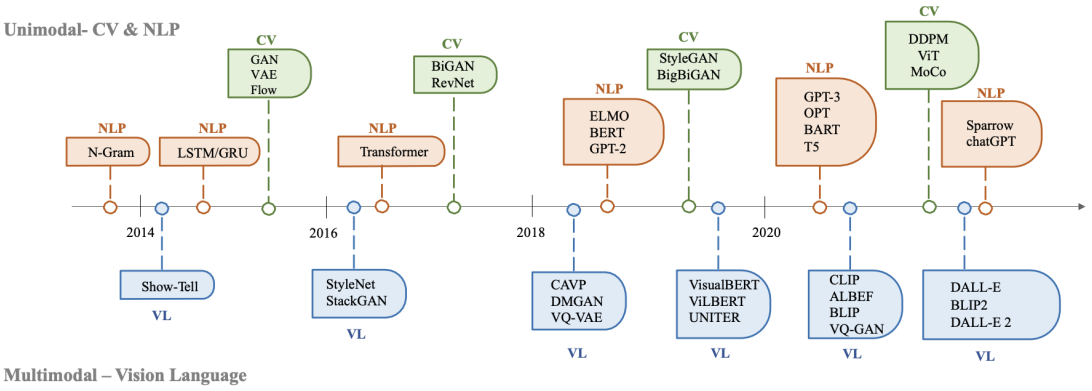

Рис. 1.0. Історія генеративного ШІ в CV, НЛП та VL.

1.2. Основні алгоритми та методи ГШІ

Нейронні мережі. Нейронні мережі - це тип машинного навчання, який моделює людський мозок. Нейронні мережі можуть використовуватися для генерування різних типів контенту, таких як текст, зображення та музика.

Машинне навчання. Машинне навчання - це галузь штучного інтелекту, яка займається розробкою алгоритмів, які можуть навчатися на даних. І воно широко застосовується в ГШІ для навчання генеративних моделей.

Глибоке навчання. Це підгалузь машинного навчання, яка використовує нейронні мережі з великою кількістю шарів. Глибоке навчання дозволяє розробити більш складні і ефективні генеративні моделі, які можуть створювати високоякісний контент.

Нейронні мережі - це тип машинного навчання, який моделює людський мозок. Нейронні мережі складаються з вузлів, які називаються нейронами. Нейрони пов'язані між собою зв'язками, які називаються синапсами.

Нейронні мережі можуть навчатися на даних. Навчання нейронної мережі відбувається шляхом зміни вагів зв'язків між нейронами. Ваги зв'язків визначають, як інформація передається між нейронами.

Нейронні мережі можуть використовуватися для генерування різних типів контенту, таких як текст, зображення та музика.

Machine Learning Машинне навчання - це галузь штучного інтелекту, яка займається розробкою алгоритмів, які можуть навчатися на даних. Машинное навчання широко застосовується в ГШІ для навчання генеративних моделей. Машинне навчання можна розділити на два основних типи: supervised learning та unsupervised learning.

Supervised learning - це тип машинного навчання, в якому модель навчається на наборі даних, який містить як вхідні дані, так і бажані вихідні дані. Модель вчиться знаходити зв'язок між вхідними даними та бажаними вихідними даними, щоб потім використовувати цей зв'язок для прогнозування вихідної величини для нових даних.

Supervised learning - це тип машинного навчання, в якому модель навчається на наборі даних, який містить як вхідні дані, так і бажані вихідні дані. Модель вчиться знаходити зв'язок між вхідними даними та бажаними вихідними даними, щоб потім використовувати цей зв'язок для прогнозування вихідної величини для нових даних.

Supervised learning є одним з найпоширеніших підходів до машинного навчання. Він використовується в широкому спектрі задач, включаючи

Класифікацію для визначення, до якої категорії належить дана одиниця даних та Регресію для прогнозування значення вихідної величини для даних.

Supervised learning вимагає набору даних, який містить як вхідні дані, так і бажані вихідні дані. Якість прогнозів, які виробляє модель supervised learning, залежить від якості даних, на яких вона була навчена.

Наприклад, модель supervised learning може бути навчена на наборі даних зображень автомобілів та мотоциклів. Набір даних містить як зображення автомобілів, так і зображення мотоциклів, а також інформацію про те, до якої категорії належить кожне зображення. Модель може навчитися знаходити зв'язок між характеристиками зображення (наприклад, формою, розміром, кольором) та категорією, до якої воно належить. Після навчання модель можна використовувати для прогнозування, чи є дане зображення автомобілем чи мотоциклом.

Supervised learning є потужним інструментом, який можна використовувати для вирішення широкого спектру задач. Однак він вимагає набору даних з високою якістю, щоб забезпечити точність прогнозів.

гнозування ціни на акції або для прогнозування продажів.

Supervised learning вимагає набору даних, який містить як вхідні дані, так і бажані вихідні дані. Якість прогнозів, які виробляє модель supervised learning, залежить від якості даних, на яких вона була навчена.

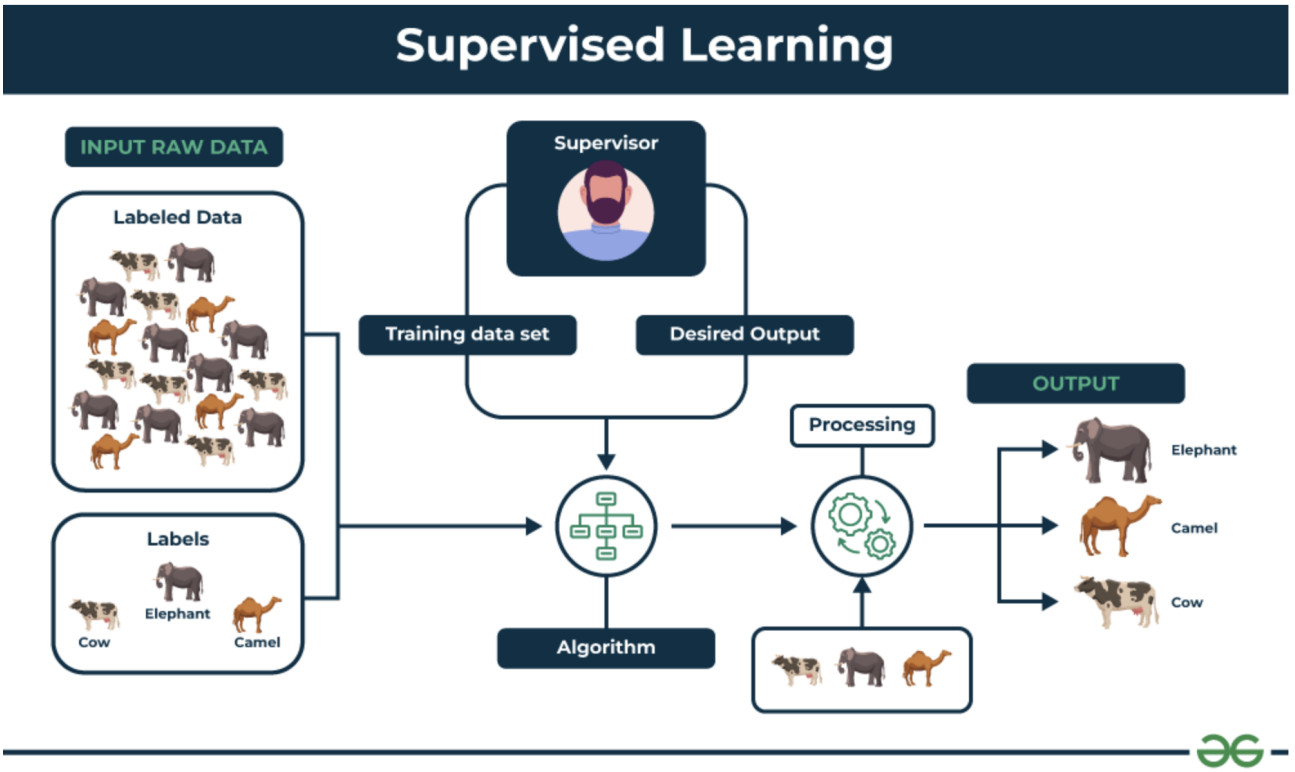

Рис. 1.1. Supervised Learning

Unsupervised learning - це тип машинного навчання, в якому модель навчається на наборі даних, який не містить бажаних вихідних даних. Модель вчиться знаходити приховані закономірності в даних, щоб потім використовувати ці закономірності для нових даних.

Unsupervised learning використовується в широкому спектрі задач, включаючи:

1. Кластеризація - поділ даних на групи, які мають спільні характеристики. Наприклад, модель unsupervised learning може бути навчена на наборі даних зображень автомобілів, щоб об'єднати зображення автомобілів, які мають схожі характеристики, такі як форма, розмір, колір.

2. Дименсійне скорочення - зменшення кількості вимірів у наборі даних, не втрачаючи при цьому важливої інформації. Наприклад, модель unsupervised learning може бути навчена на наборі даних зображень, щоб зменшити кількість пікселів у кожному зображенні, не втрачаючи при цьому основних характеристик зображення.

3. Аномалійне виявлення - виявлення даних, які відрізняються від інших даних у наборі. Наприклад, модель unsupervised learning може бути навчена на наборі даних фінансових даних, щоб виявити потенційні шахрайські операції.

Unsupervised learning не вимагає набору даних з бажаними вихідними даними. Якість результатів unsupervised learning залежить від якості даних, на яких модель була навчена.

Unsupervised learning є потужним інструментом, який можна використовувати для вирішення широкого спектру задач. Однак він може бути складнішим для розуміння та застосування, ніж supervised learning.

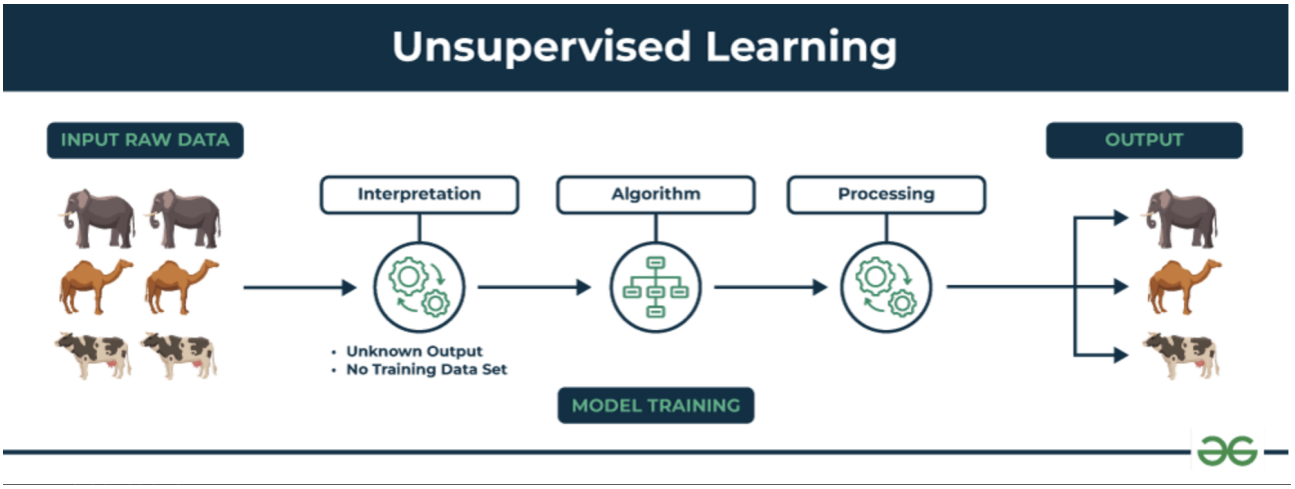

Рис. 1.2. Unsupervised Learning

Deep Learning - це підгалузь машинного навчання, яка використовує нейронні мережі з великою кількістю шарів. Глибоке навчання дозволяє розробити більш складні і ефективні генеративні моделі, які можуть створювати високоякісний контент.

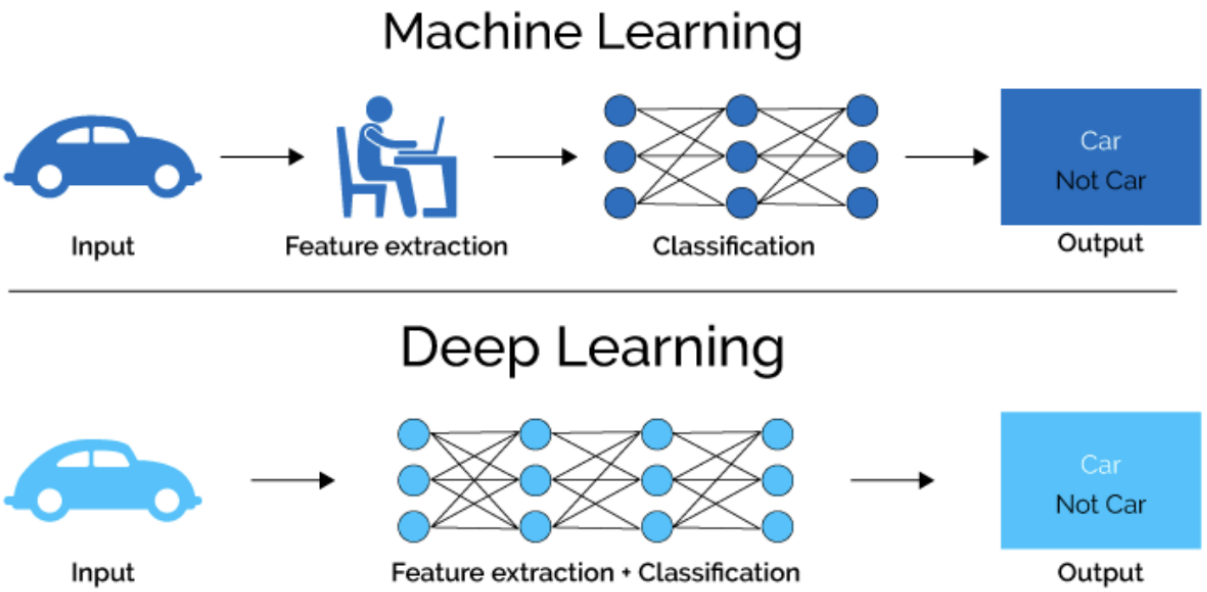

Рис. 1.3. Як працює глибоке навчання та машинне навчання

Одним з найважливіших досягнень в галузі глибокого навчання в контексті ГШІ було створення моделі Generative Adversarial Network (GAN) в 2014 році. GAN - це тип генеративної моделі, яка використовує два типи нейронних мереж: генератор і дискримінатор. Генератор створює новий контент, а дискримінатор намагається відрізнити створений контент від справжнього.

GAN дозволили значно підвищити якість генерованого контенту. Наприклад, GAN можна використовувати для генерації реалістичних зображень людей, тварин, об'єктів та сцен.

Інші методи ГШІ: Окрім нейронних мереж та машинного навчання, в ГШІ також застосовуються інші методи, такі як:

Стохастичне моделювання. Стохастичне моделювання використовується для генерації випадкових даних, які відповідають деяким заданим характеристикам.

Методи на основі штучного інтелекту. В ГШІ також застосовуються методи на основі штучного інтелекту, такі як генетичні алгоритми, евристичний пошук та інші.

1.3. Типи Генеративних Моделей у ШІ

Transformers або Трансформери (наприклад, GPT-3) - це клас нейромережевих архітектур, які здійснили революцію в галузі обробки природної мови (NLP) і не тільки. На відміну від традиційних рекурентних нейронних мереж (RNN), які обробляють дані послідовно, трансформатори покладаються на механізм уваги для вивчення довгострокових залежностей між елементами вхідної послідовності. Це дозволяє їм досягати найсучаснішої продуктивності в різних завданнях НЛП, включаючи генерацію тексту, машинний переклад, відповіді на запитання тощо.

Ключ до розуміння трансформерів лежить у концепції уваги. Замість того, щоб обробляти елементи вхідної послідовності один за одним, механізм уваги дозволяє моделі зосередитися на найбільш релевантних частинах вхідних даних на основі поточного контексту. Це дозволяє моделі вловлювати довгострокові залежності та взаємозв'язки між елементами більш ефективно, ніж VAE, які часто обмежені проблемою зникаючого градієнта.

Трансформери в дії:

Генерація тексту: Трансформери, такі як GPT-3/4, чудово генерують текст людської якості, включаючи художні тексти, сценарії, вірші і навіть музичні твори. Вони можуть адаптуватися до різних стилів і жанрів, що робить їх цінними інструментами для створення контенту.

1. Автоматичне узагальнення тексту: Трансформери можуть ефективно узагальнювати довгі фрагменти тексту, виокремлюючи ключову інформацію та створюючи стислі резюме. Це допомагає користувачам швидко зрозуміти основні моменти довгих статей або документів.

2. Відповіді на запитання: Трансформери чудово відповідають на відкриті, складні та фактичні запитання, що базуються на певному контексті. Вони можуть аналізувати складні взаємозв'язки в тексті та надавати вичерпні й інформативні відповіді.

3. Генерація коду: Трансформери все частіше використовуються для генерації коду, автоматизації повторюваних завдань і допомоги програмістам. Вони можуть навчатися на існуючих базах коду та генерувати фрагменти коду на основі конкретних вимог.

4. Написання музики: Трансформери навіть показали себе у написанні музики, створюючи твори з різними стилями та настроями. Вони можуть вчитися на існуючих музичних творах і створювати унікальні композиції, які відповідають музичним правилам і закономірностям.

5. За межами НЛП: Застосування Трансформерів виходить за рамки НЛП. Їх застосовували для вирішення різних завдань в інших сферах, зокрема

6. Розпізнавання зображень: Трансформери можна використовувати для аналізу зображень і створення підписів або описів.

7. Прогнозування часових рядів: Аналізуючи минулі дані, трансформатори можуть передбачати майбутні тенденції та закономірності.

8. Виявлення ліків: Трансформери можуть допомогти визначити потенційних кандидатів у ліки, аналізуючи великі набори даних про хімічні сполуки.

Виклики та майбутні напрямки:

Незважаючи на свої успіхи, трансформери все ще стикаються з проблемами:

Обчислювальні витрати: Навчання великих моделей трансформерів вимагає значних обчислювальних ресурсів.

Тлумачення: Розуміння того, як трансформери приймають рішення та інтерпретують інформацію, яку вони обробляють, залишається постійним викликом.

Упередженість: Як і будь-яка інша модель ШІ, Трансформери можуть успадковувати упередженість від даних, на яких вони навчаються. Зменшення упередженості та забезпечення справедливості має вирішальне значення.

Дослідники активно працюють над вирішенням цих проблем і вивчають нові можливості застосування трансформерів. Деякі перспективні майбутні напрямки включають:

Розробка більш ефективних архітектур: Оптимізація архітектур Трансформерів для швидшого навчання та висновків.

Вдосконалення методів інтерпретації: Розробка нових методів для розуміння внутрішньої роботи Трансформерів і процесів прийняття ними рішень.

Вивчення нових застосувань: Застосування трансформерів у ширшому діапазоні сфер, ніж НЛП, розширюючи межі можливого за допомогою ШІ.

Генеративні змагальні мережі (GAN) - це потужний і універсальний клас моделей машинного навчання, який став гарячою темою в останні роки. Вони особливо відомі своєю здатністю генерувати високореалістичні зображення та інші форми творчого контенту. У цій статті ми заглибимося у захопливий світ GAN, досліджуючи їхню внутрішню роботу, застосування та майбутній потенціал.

Генеральні мережі складаються з двох основних компонентів:

Генератор: Ця нейронна мережа діє як творча сила. Вона приймає випадковий шум як вхідні дані і виробляє синтетичні дані, які намагається не відрізнити від реальних даних.

Дискримінатор: Ця нейромережа діє як критик. Вона аналізує як реальні, так і згенеровані дані, намагаючись відрізнити одні від інших.

Ключ до успіху GAN полягає в протиборстві між цими двома компонентами. Генератор постійно намагається покращити свою здатність обдурити дискримінатор, створюючи більш реалістичні результати. У свою чергу, дискримінатор постійно вдосконалює свої навички, щоб краще розрізняти реальні та фейкові дані. Ця безперервна конкуренція спонукає обидві мережі вчитися і вдосконалюватися, що в кінцевому підсумку призводить до створення все більш якісних синтетичних даних.

GAN знайшли безліч застосувань у різних сферах, зокрема

1. Генерація зображень: Створення реалістичних зображень облич, об'єктів і навіть сцен, включаючи фотореалістичне редагування і покращення.

2. Створення творів мистецтва: Створення нових креативних мистецьких стилів, від картин до скульптур і музики.

3. Генерація тексту: Створення реалістичних і зв'язних текстових форматів, зокрема віршів, коду, скриптів і навіть новинних статей.

4. Розширення даних: Штучне розширення існуючих наборів даних для покращення продуктивності моделей машинного навчання.

5. Створення відеоігор: Створення реалістичних текстур, персонажів і середовищ для захопливого ігрового досвіду.

6. Медична візуалізація: Синтез реалістичних медичних зображень для навчання та розробки діагностичних інструментів.

7. Відкриття ліків: Генерування нових молекул з бажаними властивостями для розробки ліків.

Незважаючи на свої вражаючі можливості, ГАН все ще стикаються з певними проблемами. До них відносяться:

1. Нестабільність навчання: Змагальний процес може призвести до нестабільного навчання, що ускладнює досягнення оптимальних результатів.

2. Колапс режиму: Генератор може застрягти в локальному оптимумі, генеруючи лише обмежену кількість результатів.

3. Відсутність інтерпретованості: Може бути важко зрозуміти, як GAN приймають рішення, що перешкоджає їх подальшому розвитку та застосуванню.

Дослідники активно працюють над вирішенням цих проблем і розширенням можливостей ШІ. Деякі перспективні майбутні напрямки включають:

– розробка нових архітектур: Дослідження нових мережевих архітектур, які є більш стабільними та ефективними.

– включення попередніх знань: Використання існуючих знань для спрямування генератора на створення конкретних типів даних.

Методи інтерпретованості: Розробка методів для розуміння того, як вони приймають рішення, та сприяння кращому контролю за їхніми результатами.

ГАН - це потужна і новаторська технологія, що має потенціал для революції в різних галузях. Оскільки дослідження продовжуються, ми можемо

очікувати на появу ще більш приголомшливих та інноваційних застосувань, що розширюють межі можливого за допомогою штучного інтелекту.

Варіаційні автокодери (VAE) - це потужний клас генеративних моделей у сфері глибокого навчання. У той час як традиційні автокодери зосереджені на стисненні та реконструкції даних, VAE роблять крок далі, вивчаючи приховане просторове представлення даних, що дозволяє їм генерувати нові та схожі точки даних.

VAE складається з двох основних компонентів:

1. Кодер: Ця нейронна мережа стискає вхідні дані в низьковимірне представлення латентного простору. Замість того, щоб просто надавати одноточкову оцінку, VAE виводить розподіл ймовірностей, фіксуючи притаманну даним мінливість.
2. Декодер: Отримавши вибірку з латентного простору, ця нейронна мережа реконструює вихідну точку даних. Декодер має на меті генерувати дані, які дуже схожі на вхідні, а також фіксувати основні характеристики розподілу даних.

VAE відрізняються від стандартних автокодерів кількома ключовими особливостями:

1. Імовірнісне представлення: VAE включають імовірнісні міркування у свій латентний простір, що дозволяє їм генерувати нові та різноманітні точки даних у межах вивченого розподілу.
2. Генеративні можливості: На відміну від традиційних автокодерів, вони можуть генерувати нові точки даних, подібні до тих, які вони бачили під час навчання. Це відкриває широкі можливості для застосування в різних галузях.

Варіаційні автокодери знайшли різноманітне застосування в різних галузях:

Генерування зображень: VAE можна використовувати для генерування реалістичних зображень облич, об'єктів і навіть абстрактних візерунків, часто перевершуючи за якістю традиційні генеруючі моделі.

1. Стиснення даних: VAE можуть ефективно стискати дані, зберігаючи лише представлення латентного простору, що може бути корисним для таких задач, як стиснення зображень та аудіо.

2. Покращення зображень: VAE можна використовувати для зашумлення зображень, надвисокої роздільної здатності зображень і навіть для зафарбовування відсутньої інформації, що призводить до покращення якості зображень.

3. Машинний переклад: VAE можна використовувати для перекладу тексту між різними мовами, вивчаючи спільне представлення латентного простору, яке фіксує семантичне значення тексту.

4. Генерація музики: VAE можна використовувати для створення нових музичних творів, вивчаючи основну структуру і закономірності існуючої музики, що дозволяє створювати творчі музичні композиції.

5. Виклики та майбутні напрямки: Хоча VAE пропонують значні переваги, вони також стикаються з певними проблемами:

6. Складність навчання: Навчання VAE може бути пов'язане з великими обчислювальними витратами і вимагає ретельного налаштування гіперпараметрів для досягнення оптимальної продуктивності.

7. Колапс режиму: Подібно до GAN, VAE можуть страждати від колапсу режиму, коли вони генерують лише обмежену кількість результатів, незважаючи на те, що базовий розподіл даних є більш різноманітним.

8. Інтерпретованість: Розуміння того, як VAE приймають рішення та інтерпретують латентний простір, залишається активною сферою досліджень.
Забігаючи наперед, дослідники активно працюють над вдосконаленням VAE шляхом

Розробки нових архітектур: Вивчення нових архітектур, які є більш ефективними і стабільними, особливо для великих обсягів даних.

Варіаційні фреймворки: Впровадження нових варіаційних методів, які пропонують кращий контроль над латентним простором і підвищують якість згенерованих даних.

Методи інтерпретації: Розробка нових методів для розуміння внутрішньої роботи VAE та отримання уявлення про процес прийняття ними рішень.

Висновок: VAE пропонують потужний і універсальний підхід до побудови генеративних моделей, які відображають суть складних даних. З розвитком досліджень і вирішенням проблем можна очікувати, що VAE відіграватимуть дедалі важливішу роль у різних сферах, розширюючи межі можливого за допомогою ШІ.

Авторегресійні моделі.

Опис: Авторегресійні моделі використовують попередні точки даних для прогнозування наступних точок. Вони генерують дані послідовно, одну точку за іншою.

Застосування: Ці моделі широко використовуються в генерації тексту, музики, мовних моделей та в аналізі часових рядів.

Генеративні моделі на основі випадкового лісу

Опис: Використовують ансамбль дерев рішень для моделювання розподілу даних та генерації нових даних, які відповідають цьому розподілу.

Застосування: Застосовуються для класифікації, регресії та генерації даних у різних сферах.

Глибокі баєсові мережі

Опис: Ці мережі використовують принципи баєсової статистики та глибокого навчання для генерації даних. Вони поєднують статистичні моделі з глибокими нейронними мережами.

Застосування: Використовуються в задачах передбачення, моделювання невизначеності та рішення складних проблем, де потрібно моделювати ймовірності.

Піксельні Рекурентні Нейронні Мережі (PixelRNN)

Опис: PixelRNN використовують рекурентні нейронні мережі для послідовного створення зображень піксель за пікселем.

Застосування: Особливо ефективні у генерації зображень та відео, а також у завданнях, де необхідно детально контролювати кожен аспект візуального виводу.

Sequence-to-Sequence Models (Seq2Seq)

Опис: Використовують дві RNN: одну для читання вхідної послідовності даних (енкодер) та іншу для генерації нової послідовності (декодер).

Застосування: Машинний переклад, автоматизовані системи відповідей, генерація тексту.

Кожен з цих типів генеративних моделей має свої унікальні особливості та специфічні застосування, що робить їх важливими для розуміння та застосування генеративного ШІ у різних сферах

1.3. Перспективи та Майбутнє Генеративного ШІ

Майбутнє Генеративного Штучного Інтелекту (ГШІ) розкриває перед собою безмежний потенціал та відкриває нові можливості для розвитку технологій та впливу на суспільство. Прогнозування майбутнього розвитку ГШІ включає наступні аспекти:

1. Зростання Вище та Глибше**:** Ми можемо очікувати подальше зростання розміру та глибини генеративних моделей. Майбутні версії ГШІ будуть мати більше параметрів та здатність генерувати більш складний та контекстуальний контент.

2. Розширення Сфер Застосування**:** ГШІ знайде нові сфери застосування, включаючи виробництво, освіту, наукові дослідження, екологію, та багато інших. Він стане необхідним інструментом у багатьох професійних галузях.

3. Етичні та Юридичні Аспекти**:** Розвиток ГШІ також підніме багато етичних і юридичних питань. Суспільство буде ставити питання про власність та використання генерованого контенту, а також про захист приватності та етичність використання ГШІ.

4. Дослідження Узагальнених Моделей**:** Потенційно нові напрямки досліджень включають створення узагальнених моделей, здатних працювати в різних сферах та навчатися на різних завданнях без великої кількості даних.

5. Синтез та Креативність**:** ГШІ може розвиватися в напрямку здатності до синтезу різних типів інформації та створення нових знань. Він також може стати більш креативним, сприяючи творчому процесу в різних сферах.

6. Соціокультурний Вплив**:** ГШІ буде мати значущий соціокультурний вплив, включаючи зміни в способах комунікації, мистецтві та культурі. Він може допомагати вирішувати важливі глобальні проблеми.

Загалом, майбутнє Генеративного Штучного Інтелекту обіцяє бути захопливим та перетворюючим для суспільства та технологій. Потенційні нові напрямки досліджень вказують на можливості розвитку цієї галузі, яка продовжує прокладати шляхи для нових відкриттів та інновацій.

1.4. Використання генеративного ШІ в додатку

Chat GPT-4, будучи однією з найсучасніших моделей на базі трансформерів, відіграє ключову роль у "Ideations". Ця модель використовується для інтерпретації та перетворення текстових запитів користувачів на детальні описи сцен. Такий підхід дозволяє користувачам виражати свої ідеї у вільній формі, після чого Chat GPT-4 структурує та уточнює ці описи для подальшої візуалізації.

Модель Chat GPT-4 має мільярди параметрів, що робить її надзвичайно потужною у генерації тексту та розумінні запитів користувача. Велика кількість параметрів дозволяє моделі "навчитися" на великому обсязі текстових даних і засвоїти різноманітні лінгвістичні відмінності та контексти. Це робить модель більш адаптивною до різних мовних завдань та стилів взаємодії.

Проте велика кількість параметрів також вимагає значних розрахункових можливостей. Такі моделі потребують великих обчислювальних ресурсів для

навчання та інференсу. Це може включати в себе використання потужних серверів або спеціалізованих обчислювальних кластерів. Крім того, велика кількість параметрів може призвести до значного споживання енергії під час роботи, що може бути важливим аспектом у питаннях сталий розвиток та витрати на обслуговування.

Застосування Stable Diffusion для створення візуальних зображень

Основи Stable Diffusion: Stable Diffusion є однією з передових технологій у галузі генеративного штучного інтелекту, спрямованою на створення візуальних зображень. Вона використовує принципи глибокого навчання, особливо архітектуру, засновану на генеративно-змагальних мережах (GAN), для перетворення текстових описів на деталізовані візуальні зображення.

Принцип Роботи Stable Diffusion

– Прийом Текстових Описів: Stable Diffusion отримує текстові описи, сформульовані за допомогою Chat GPT-4, які можуть включати деталі щодо кольорів, форм, композиції та тематики сцен.

– Генерація Зображення: На основі цих описів, Stable Diffusion генерує відповідні зображення, використовуючи величезні набори даних для створення реалістичних та якісних візуалізацій.

– Візуальна Когерентність: Модель здатна забезпечувати візуальну когерентність та стилістичну цілісність зображень, що робить її ідеальною для створення складних візуальних сцен.

Переваги Використання Stable Diffusion в "Ideations"

– Ефективність та Швидкість: Здатність швидко генерувати високоякісні зображення зі складних текстових описів.

– Гнучкість: Можливість адаптуватися до широкого спектра стилів та вимог до дизайну.

– Креативність: Стимулювання креативного процесу, дозволяючи користувачам експериментувати з різними візуальними концепціями.

1.4.3. Технічні Аспекти та Виклики

Обробка Великих Даних: Застосування потужних алгоритмів для обробки великих наборів даних та забезпечення точності зображень.

Точність та Реалістичність: Підтримка високої точності та реалістичності генерованих зображень, що є ключовим для задоволення вимог користувачів.

Інтерпретація Описів: Здатність правильно інтерпретувати та візуалізувати складні та абстрактні описи.

Майбутні Перспективи

– Покращення Алгоритмів: Неперервне оновлення та вдосконалення алгоритмів для забезпечення кращої якості та швидкості генерації зображень.
– Адаптація до Нових Вимог: Гнучкість у внесенні змін та адаптації до нових трендів та вимог у сфері візуального дизайну.
– Інтеграція та взаємодія компонентів

Інтеграція Chat GPT-4 та Stable Diffusion у "Ideations" є взірцевим прикладом використання генеративного ШІ для створення комплексного продукту. Ця інтеграція демонструє, як різні аспекти ШІ можуть бути об'єднані для створення інструменту, який переводить людські ідеї та мову в візуальні образи. Такий підхід не тільки сприяє креативності, але й вносить значні поліпшення в процес пре-продакшену, знижуючи часові та ресурсні витрати.

Майбутнє застосування генеративного ШІ в "Ideations"

В майбутньому, "Ideations" може розширити свої можливості, інтегруючи новітні розробки в області генеративного ШІ. Це може включати покращення точності генерації зображень, більшу кастомізацію візуалізацій та розширення функціональності для різних типів медіаконтенту.

ВИСНОВКИ ДО РОЗДІЛУ 1

У цьому розділі було проведено глибокий аналіз та огляд теоретичних основ генеративного штучного інтелекту (ГШІ), його ключових алгоритмів, історії розвитку та різноманітності моделей. Дослідженню ролі цих технологій у різних сферах застосування ШІ, що дало не лише ґрунтовну базу знань, але й інструментарій для осмислення й вибору оптимальних рішень щодо технологій ШІ для розробки веб-додатку "Ideations".

Увага була приділена можливостям генерації тексту та зображень, швидкості обробки, актуальності технологій, архітектурі та потенціалу до масштабування та розширення. Це надзвичайно важливо, оскільки вибір адекватних технологій впливає не тільки на поточну ефективність та функціональність додатку, але й на його довгостроковий розвиток. Отримане розуміння динаміки розвитку в області ШІ дозволило вибрати найбільш відповідні та надійні моделі, які задовольняють потреби проекту та відкривають перспективи для майбутніх інновацій.

Крім того, у розділі детально описано конкретні моделі, які були використані у роботі. Вибір кожної з моделей обґрунтовано на основі здатності до генерації високоякісних зображень, ефективності взаємодії з текстовими описами, і можливості адаптації під специфічні завдання проекту "Ideations". Цей підхід забезпечує не тільки технічну ефективність, але й творчу гнучкість, що є критично важливим для розвитку інноваційних технологічних продуктів.

У підсумку, дана частина роботи не лише забезпечила необхідну теоретичну підготовку для ефективної реалізації проекту "Ideations", але й створила міцний фундамент для майбутніх наукових та практичних пошуків у сфері штучного інтеле

РОЗДІЛ 2
ПРОЕКТУВАННЯ ЗАСТОСУНКУ

2.1. Планування розробки та вимог

На етапі аналізу вимог проведено ретельний розгляд та збір інформації від потенційних користувачів та стейкхолдерів, щоб визначити ключові потреби і вимоги до додатку "Ideations". Цей етап включав в себе такі основні дії:

1. Збір Інформації від Користувачів: Було проведено інтерв'ю та опитування з потенційними користувачами додатку, а також вивчено їхні запити та потреби. Важливою була взаємодія з майбутніми користувачами, щоб зрозуміти, як вони планують використовувати додаток "Ideations" та які завдання вони намагаються вирішити за його допомогою. Однією з ключових потреб була потреба в швидкому та ефективному створенні візуального контенту на основі текстових промптів.

2. Формулювання Функціональних та Нефункціональних Вимог: На основі інформації, зібраної від користувачів та стейкхолдерів, були сформульовані функціональні та нефункціональні вимоги до додатку "Ideations". Функціональні вимоги визначають, які конкретні функції та можливості має мати додаток, зокрема, можливість генерації візуального контенту на основі введеного тексту.

3. Пріорітизація Вимог: Вимоги були ранжировані за важливістю та пріоритетом, щоб визначити, які функції та можливості мають бути реалізовані в першу чергу, а які можуть бути відкладені на пізніші етапи розробки.Функціональні вимоги того що має мати застосунок:

Система повинна мати можливість генерувати розкадровки та скетчі на основі описів сцен, які надаються користувачем у текстовому форматі. Користувач повинен мати змогу ввести докладний опис кожної сцени, включаючи діалоги, події, рух персонажів та інші деталі. Система повинна враховувати цей текстовий опис та на його основі автоматично створювати відповідні розкадровки та скетчі.

Користувач повинен мати можливість редагувати текстовий опис під кожною сценою для внесення змін або уточнень. Це дозволить користувачу налаштовувати деталі сцен та адаптувати їх до своїх потреб. Після внесення змін у текстовий опис, система повинна надавати можливість перегенерувати відповідні розкадровки та скетчі відповідно до оновленого опису.

Такий функціонал дозволить користувачам створювати та налаштовувати сцени для анімаційних проектів за допомогою текстового опису, що спростить та прискорить процес створення візуального контенту.

Редагування та Перегляд Сцен:

Система повинна надавати користувачам повний контроль над згенерованими сценами. Користувачі повинні мати можливість редагувати сцени, вносячи зміни до об'єктів, додавати нові об'єкти, видаляти існуючі, а також змінювати їх позиції, розміри та інші параметри. Це дозволить користувачам налаштовувати кожну сцену з точністю до деталей, гарантуючи повну індивідуалізацію створеного візуального контенту.

Для полегшення процесу редагування сцен система повинна надавати інтуїтивний інтерфейс, який дозволить користувачам взаємодіяти з об'єктами та параметрами сцен за допомогою миші або інших зручних інструментів. Користувачі повинні мати доступ до інструментів для малювання, редагування тексту, переміщення та зміни розміру об'єктів, а також для видалення та додавання нових об'єктів на сцену.

Крім того, система повинна підтримувати режим попереднього перегляду, який дозволяє користувачам переглядати створені сцени в реальному часі. Цей

режим допоможе користувачам оцінити вигляд та характеристики сцен перед їх використанням у відеопроекті, що підвищить якість та точність результату.
Загалом, система має забезпечувати максимальну гнучкість та контроль користувача над процесом редагування та налаштування сцен для створення візуального контенту.

Інтерфейс застосунку "Ideations" повинен відповідати вимогам зручності та інтуїтивної зрозумілості для користувачів. Це означає, що дизайн і структура інтерфейсу мають бути такими, щоб користувачі могли легко орієнтуватися та використовувати всі функціональні можливості додатку без зайвих зусиль.
Основні принципи дизайну інтерфейсу повинні включати:

1. Простоту: Інтерфейс має бути максимально простим і зрозумілим. Мінімізація зайвих деталей та складних операцій сприяє легкому використанню.

2. Логічну структуру: Меню, кнопки та інші елементи керування повинні бути розташовані логічно і послідовно. Користувач повинен легко знайти необхідні функції.

3. Інтуїтивність: Користувач має розуміти, як взаємодіяти з інтерфейсом без додаткових пояснень. Використання стандартних піктограм, текстових пояснень та інших засобів сприяє інтуїтивності.

4. Ефективність: Інтерфейс повинен дозволяти користувачам виконувати завдання швидко та ефективно. Мінімізація кількості кроків для досягнення мети є важливим аспектом.

5. Адаптивність: Інтерфейс повинен бути адаптованим до різних типів пристроїв та розмірів екранів. Він має коректно відображатися на комп'ютерах, планшетах та смартфонах.

6. Можливість взаємодії через веб-інтерфейс: Користувачам повинна бути доступна можливість взаємодії з системою через веб-інтерфейс без необхідності встановлення додаткового програмного забезпечення. Це робить додаток більш доступним та зручним у використанні.

Загальна мета інтерфейсу - забезпечити користувачам зручний та ефективний інструмент для створення розкадровок та скетчів без зайвих труднощів і надавати їм можливість легко взаємодіяти з системою.

Система має бути адаптована до майбутнього розширення, та бути готовою до імплементування наступних функцій:

Збереження та Завантаження Проектів: Система повинна забезпечувати користувачів зручними та надійними інструментами для збереження та завантаження своїх проектів. Користувачі повинні мати можливість зберігати створені сцени та проекти у вигляді файлів або у внутрішній базі даних системи.

Збережені проекти повинні зберігати всі внесені користувачем зміни, включаючи тексти, зображення, параметри сцен та інші налаштування. При цьому система повинна надавати можливість користувачам надавати своїм проектам назви та описи для зручності подальшого відстеження.

Користувачі також повинні мати можливість завантажувати збережені проекти для подальшої роботи над ними. Це дозволить їм відновлювати та редагувати свої проекти в майбутньому, а також ділитися ними з іншими користувачами, якщо така функціональність підтримується.

Забезпечення надійного та безпечного зберігання та завантаження проектів є важливою частиною функціональності системи, оскільки це дозволить користувачам зберігати свої досягнення та спрощувати процес роботи над великими та складними проектами.

Експорт та Друк. Система повинна надавати користувачам зручні інструменти для експорту та друку їх створених розкадровок та скетчів. Користувачі повинні мати можливість зберігати свої роботи у різних форматах, таких як зображення (наприклад, JPG, PNG), документи (PDF), а також інші популярні формати для подальшого використання або публікації.

При експорті система повинна дозволяти користувачам вибирати бажаний формат та налаштовувати параметри експорту, такі як розмір зображення, якість, роздільна здатність тощо. Це дозволить користувачам отримувати

результати в точно визначеному форматі та якості, відповідно до їх потреб та вимог.

Для друку сцен та скетчів система повинна підтримувати опції налаштування друку, такі як вибір друкованої сторінки (горизонтальна або вертикальна орієнтація), розмір аркуша, масштабування та інші параметри. Користувач повинен мати можливість попереднього перегляду друку перед виконанням операції, щоб переконатися, що результат відповідає їх очікуванням.

Функціональність експорту та друку допоможе користувачам ефективно використовувати створені сцени та скетчі для подальшого використання, представлення своїх ідей або спільного спілкування з іншими учасниками проекту.

Авторизація та Управління Користувачами. Авторизація та управління користувачами є важливою частиною функціональності системи. Вони забезпечують безпеку та індивідуалізацію досвіду користувачів.

Система повинна дозволяти користувачам авторизуватися в системі з різними ролями. Наприклад, адміністратори повинні мати розширені права доступу та можливість управляти іншими обліковими записами. Звичайні користувачі, з іншого боку, мають обмежений доступ до функціональності системи.

Крім того, система повинна надавати можливість створювати нові облікові записи користувачів та управляти ними. Це означає, що адміністратори мають можливість додавати нових користувачів, блокувати або видаляти облікові записи, а також змінювати ролі користувачів.

Ця функціональність забезпечує безпеку та зручність користування системою і робить її більш гнучкою та налаштовуваною під потреби конкретного бізнесу чи проекту.

Це лише загальний перелік функціональних вимог. Реальні вимоги можуть бути більш конкретними та деталізованими в залежності від конкретного проекту та його цілей.

2.2. Інтерфейс Користувача

Хедер (Рис. 2.0) в додатку включає в себе логотип і є важливою частиною інтерфейсу. Хоча в ньому може бути лише логотип, його наявність має декілька важливих функцій:

Брендування та Реклама в компоненті Хедер: Логотип у хедері виконує функцію брендування. Він надає ідентифікацію та візуально представляє вашу компанію, бренд чи продукт. Користувачі можуть легко впізнати вашу марку за логотипом.

1. Навігація: Хедер може містити навігаційні елементи, такі як меню або посилання, які допомагають користувачам швидко переходити до різних частин додатку чи важливих сторінок. Це спрощує взаємодію з додатком.
2. Контекст: Хедер може вказувати на контекст або функцію, що стосується поточного вмісту додатку..
3. Професійний Вигляд: Хедер додає професіоналізм та організованість до додатку. Він створює враження, що додаток доглянутий та ретельно розроблений.

Відповідність Дизайну: Логотип у хедері повинен вписуватися в загальний дизайн додатку і дотримуватися його стилю та кольорової палітри.

Отже, хедер із логотипом в додатку не лише допомагає ідентифікувати бренд, але і полегшує навігацію, надає професіональний вигляд, а також може вказувати на контекст та функціональність додатку. Ця невелика, але важлива частина інтерфейсу сприяє зручності та задоволенню користувачів.

Рис. 2.0. Вигляд компонента “Хедер”

Компонент редагування включає в себе:

Бріф, або Brief (короткий опис сценарію, засновуючи на якому відбувається генерація озвучки, та самих сцен з зображеннями) зображено на (Рис. 2.1.), (Рис. 2.2.), (Рис. 2.3.)

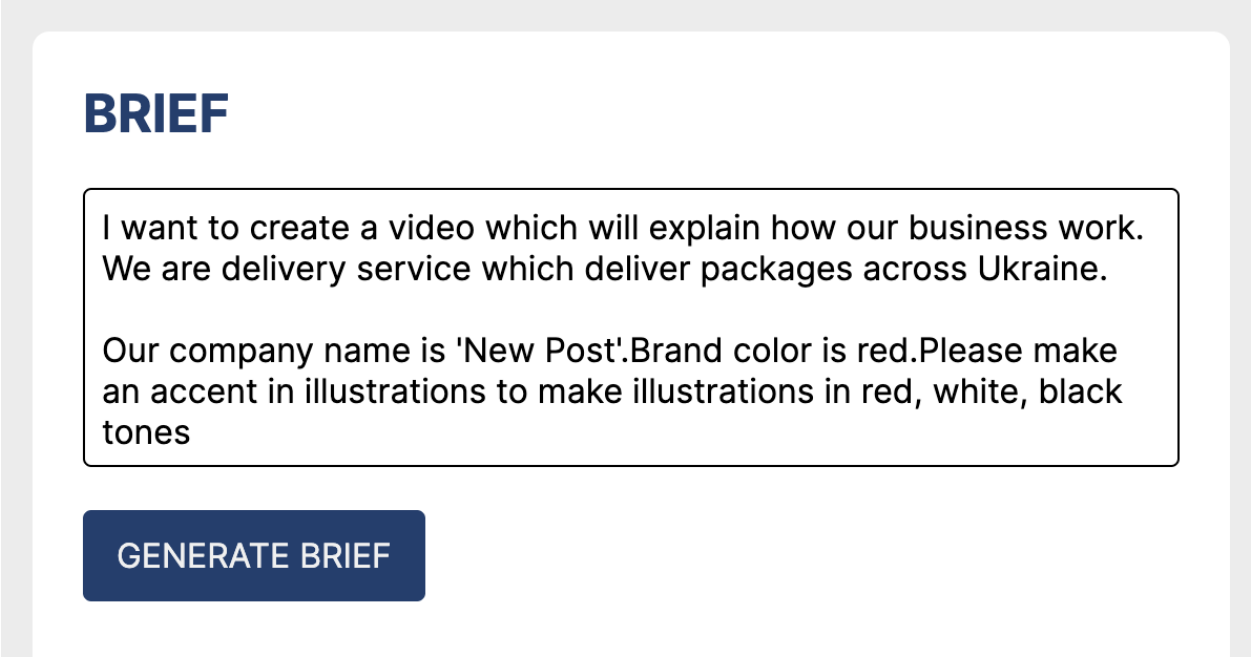

Рис. 2.1. Компонент “Бріф”

Опис відео, або Video Description (Детальний опис сцен згенерований Chat GPT4, на базі Бріфу

Рис. 2.2. Компонент “Опис відео”

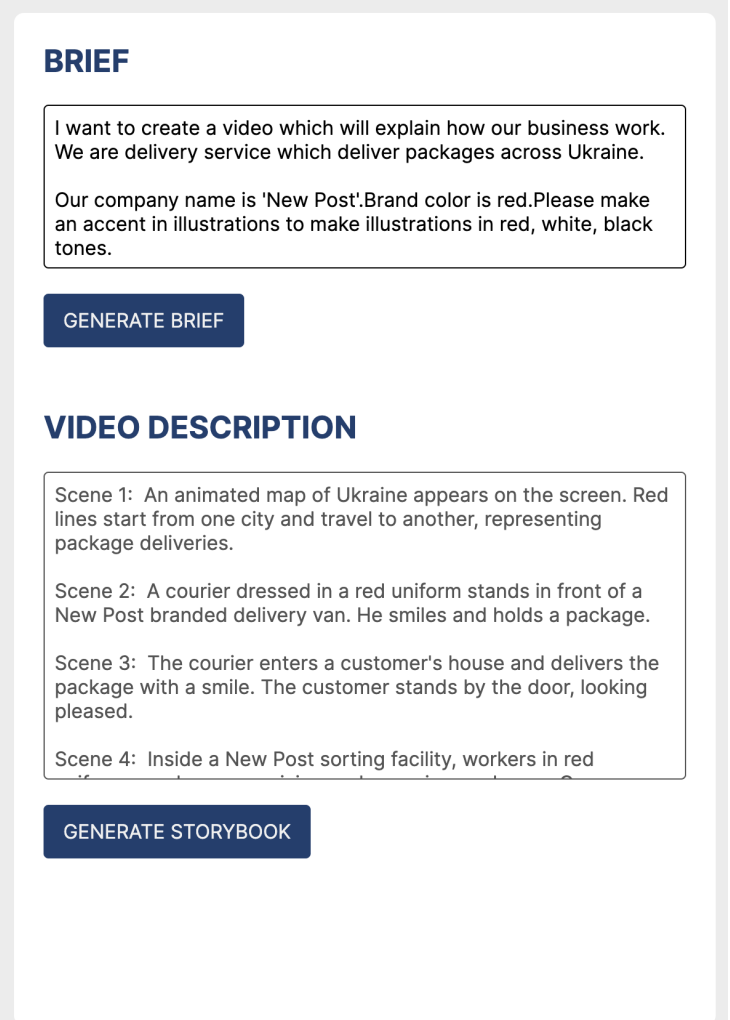

Рис. 2.3. Компоненту “Редагування”

Так як генерація зображень займає трохи часу, було вирішено покращити користувацький інтерфейс за допомогою сучасного рішення для опрацювання очікування користувача під назвою “Skeleton” або скелетні екрани.

Чим корисні скелетні екрани: Коли користувач потрапляє на сторінку додатку або веб-сайту і нічого не відбувається, він припускає, що щось не так. Щоб заохочувати позитивний користувацький досвід, важливо чітко повідомляти про те, що відбувається під час завантаження сторінки.

Іконка у вигляді дзиги може здатися очевидним вибором для повідомлення про те, що щось відбувається, або для того, щоб дати користувачеві зрозуміти, що йому потрібно зачекати. Але дослідження показують, що порожні стани без жодних ознак прогресу менш успішні в заохоченні довіри користувачів.

Скелетні екрани можна використовувати як спосіб повідомити користувачам, що у фоновому режимі відбувається якась активність - і, що дуже важливо, що це не займе багато часу. Даючи користувачам візуальні підказки про те, як довго їм доведеться чекати, перш ніж вони зможуть знову користуватися сайтом, можете допомогти побудувати довіру між користувачем і вашим продуктом.

Довгий час завантаження може дратувати користувачів, але скелетні екрани забезпечують зворотний зв'язок у ключових точках і роблять час очікування більш терпимим. Також можна використовувати скелетний екран для індикації активності. Наприклад, анімовані скелетні екрани можна використовувати як індикатор прогресу або навіть як повідомлення про помилку, яке вказує на те, що щось пішло не так. Приклад скелетного екрану зображено (Рис. 2.4.)

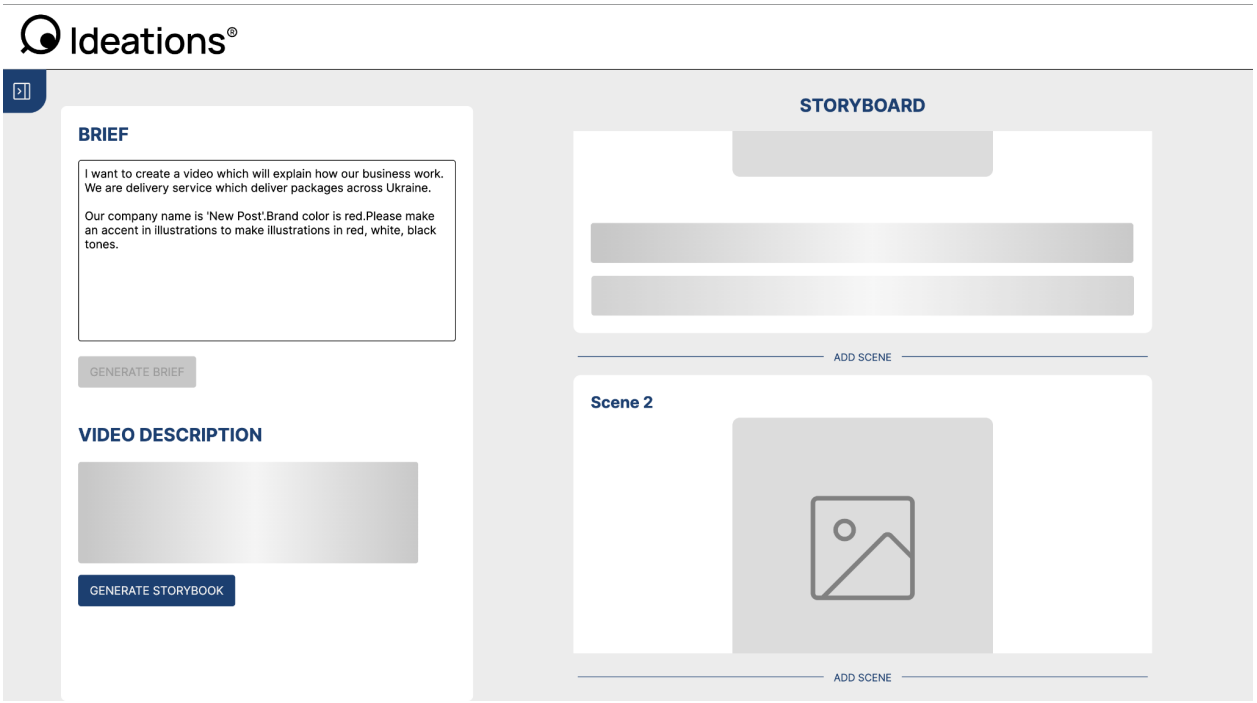

Рис. 2.4. Інтерфейс додатку під час завантаження данних

Сторіборд, складається зі сцен. Кожна сцена має

Номер, Перегляд обраного зображення, Попередні згенеровані зображення, Поле опису, Поле озвучки, Під кожною з сцен є можливість додати сцену.

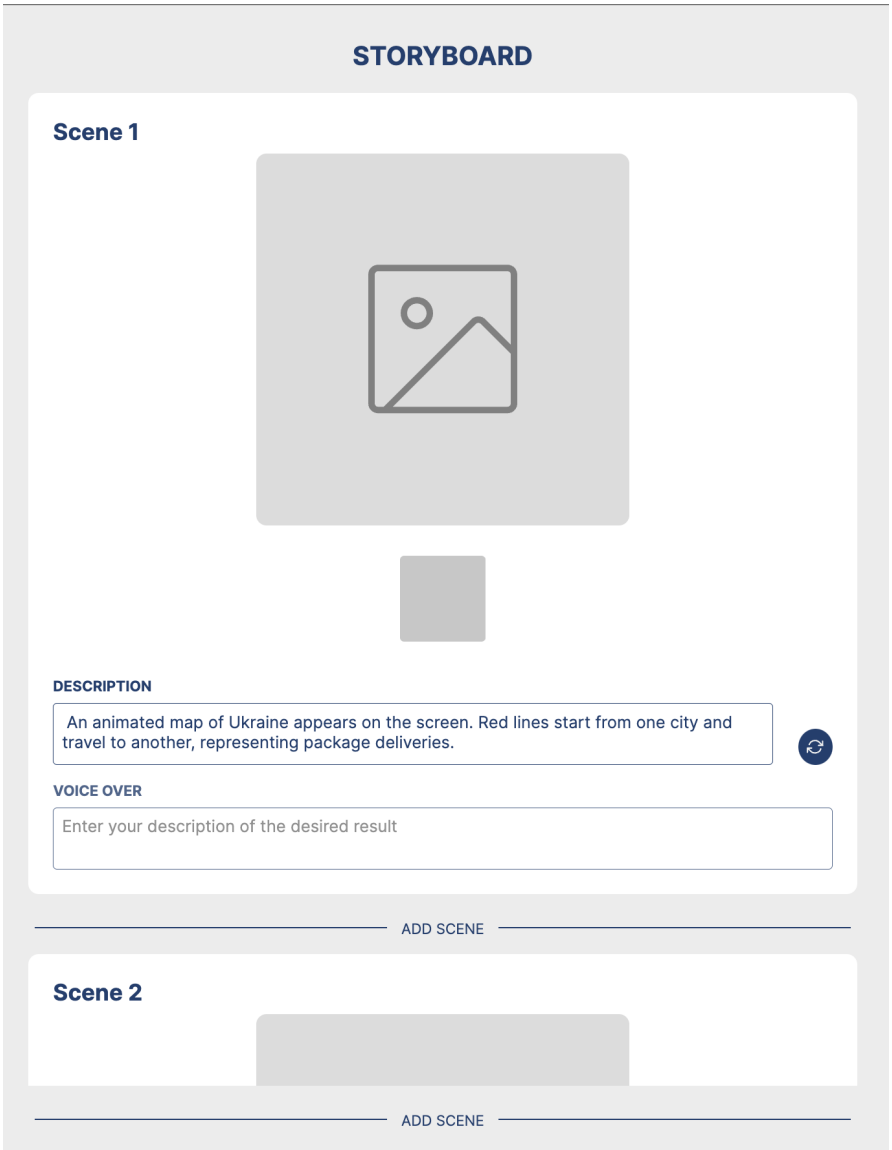

Рис. 2.5. Вигляд компоненту “Сторіборд”

Також було створено майбутній дизайн додатку (Рис. 2.6.) та (Рис. 2.7.) які включають в собі розширений функціонал та кращі, більш розвинуті функціональністю інтерфейси:

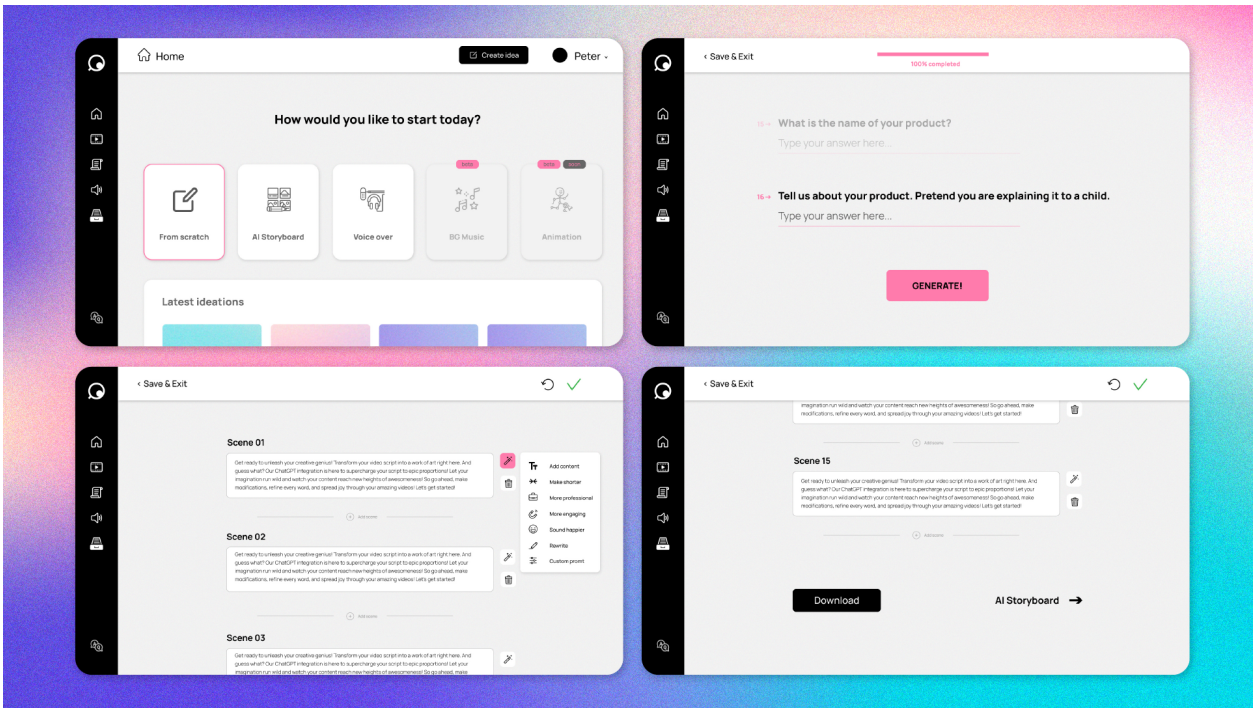

Рис. 2.6 Майбутній дизайн додатку частина 1

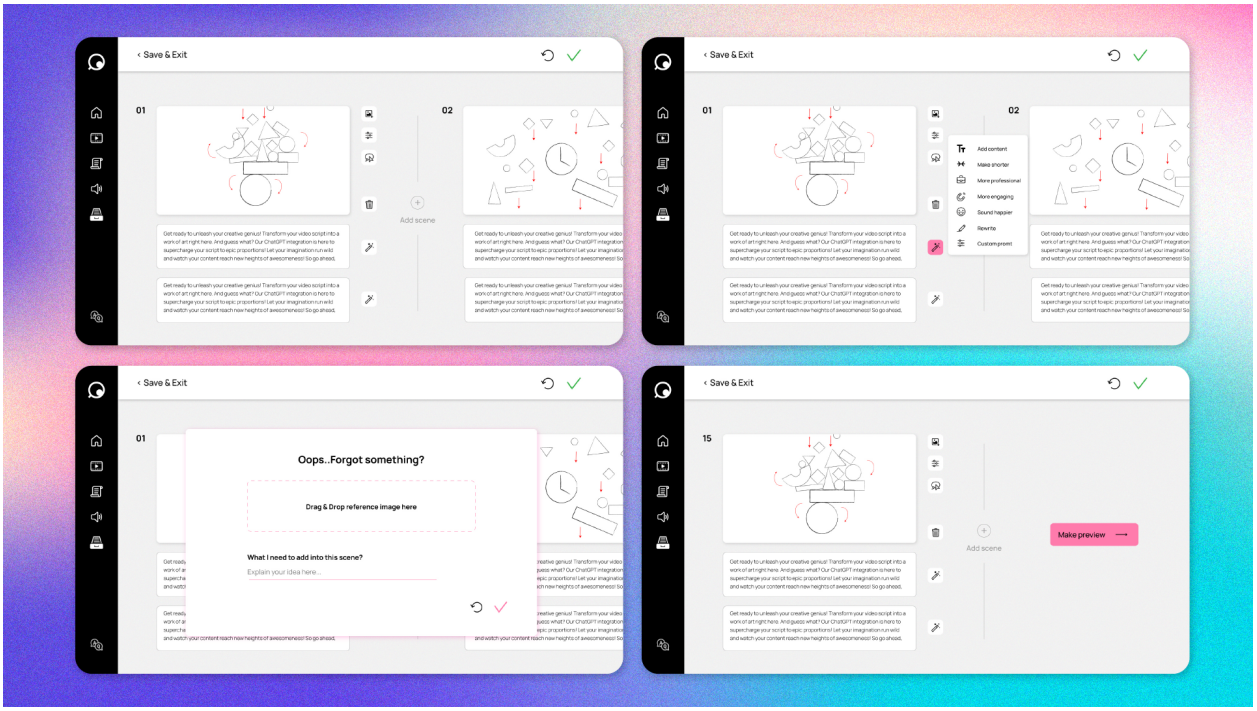

Рис. 2.7 Майбутній дизайн додатку частина 2

ВИСНОВКИ ДО РОЗДІЛУ 2

У цьому розділі була проведена детальна робота щодо вивчення потреб користувачів та їх очікувань від веб-додатку "Ideations". На основі глибоких інтерв'ю з потенційними користувачами були визначені ключові "болі" та вимоги, що дозволило сформувати функціонал додатку, який максимально відповідає їхнім потребам. Окрім технічного аспекту, увагу було приділено естетичним елементам – користувацькому інтерфейсу, використанню кольорів та брендингу, забезпечуючи, таким чином, інтуїтивно зрозуміле та привабливе візуальне оформлення.

Було детально розглянуто кожен компонент інтерфейсу, його функціонал та взаємодію з користувачем, що включало аналіз елементів навігації, віджетів, форм введення даних та інших важливих аспектів. Особливу увагу було приділено використанню скелетних екранів під час завантаження – сучасної технології, яка допомагає покращити враження користувача від взаємодії з додатком, мінімізуючи негативний вплив очікувань.

Далі, були викладені конкретні кроки користувача під час роботи з додатком, що включає в себе поетапний опис процесу від початкової реєстрації та авторизації до використання основних функцій додатку. Це дозволяє не лише показати зручність та ефективність інтерфейсу, але й наголосити на продуманості кожної деталі.

В кінці розділу було описано плани розширеного дизайну веб-додатку, який буде реалізовано в майбутньому. Цей план включає розробку нових функцій, покращення візуального оформлення та збільшення зручності користувачів. Ці плани відображають не лише бачення майбутнього розвитку додатку, але й підтверджують готовність до неперервного вдосконалення та інновац

РОЗДІЛ 3
РОЗРОБКА ЗАСТОСУНКУ

3.1. Вибір технологій

Додаток "Ideations" має модульну архітектуру, що поєднує фронтенд, розроблений на React і Typescript з використанням Vite, і бекенд на Python. Фронтенд забезпечує інтерактивний інтерфейс користувача, де вони можуть вводити бріфи для створення візуального контенту. Бекенд обробляє ці бріфи, відправляючи їх на сервер Python, де вони інтегровані з API Chat GPT-4 для генерації тексту та Stable Diffusion для створення зображень. Згенерований контент надсилається назад на фронтенд для візуалізації користувачу. Ngrok використовується для створення безпечного з'єднання між фронтендом і бекендом.

Обґрунтування вибору стку додатку Ideations

Для створення фронтенду додатку Ideations було обрано три технології: React, TypeScript, Vite, Zustand, Eslint, SCSS. Кожна з цих технологій має свої переваги та недоліки, і їхній вибір був обґрунтований з урахуванням конкретних вимог додатку.

React - це бібліотека JavaScript, розроблена компанією Facebook, яка, серед іншого, була використана для створення Instagram.com. Вона має на меті дозволити розробникам легко створювати швидкі користувацькі інтерфейси як для веб-сайтів, так і для додатків. Основною концепцією React.js є віртуальний DOM. Це дерево на основі JavaScript-компонентів, створених за допомогою React, яке імітує DOM-дерево. Воно виконує мінімальну кількість маніпуляцій з DOM, щоб підтримувати ваші React-компоненти в актуальному стані

Будучи частиною мови JavaScript, використання React має багато переваг. Продукти, створені за допомогою React, легко масштабуються, єдина мова, що використовується на стороні сервера/клієнта/мобільних пристроїв, забезпечує неперевершену продуктивність, існують шаблони робочих процесів для зручної командної роботи, код інтерфейсу користувача легко читається та підтримується, і багато іншого.

Провідні світові компанії використовують React та інші JS-технології у своїх продуктах, що визначають ринок (найяскравішими прикладами є Instagram, Reddit та Facebook).

Віртуалізуючи та зберігаючи DOM в пам'яті, React забезпечує надзвичайно швидкий рендеринг, при цьому всі зміни вигляду легко відображаються у віртуальному DOM. Спеціалізований алгоритм diff порівнює попередні та існуючі стани віртуального DOM, обчислюючи найефективніший спосіб застосування нових змін, не вимагаючи надто великої кількості оновлень. Потім вноситься мінімальна кількість оновлень для досягнення найшвидшого часу читання/запису, що призводить до загального підвищення продуктивності.

Зміни DOM уповільнюють роботу системи, а завдяки віртуалізації DOM ці зміни мінімізуються та інтелектуально оптимізуються. Всі маніпуляції з віртуальним DOM відбуваються "за лаштунками", тобто внутрішньо і автономно, що також дозволяє значно економити апаратні ресурси (наприклад, потужність процесора і заряд батареї в мобільних пристроях, що також повинно підказати вам, коли варто використовувати React.js).

Синтаксис JSX для розширеного HTML

З React.js можна використовувати декларативний синтаксис HTML безпосередньо в JavaScript-коді. Щоб показати користувацький інтерфейс, браузери декодують HTML-тексти. Вони роблять це шляхом побудови DOM-дерев, якими потім можна маніпулювати за допомогою JavaScript для створення інтерактивного інтерфейсу. Схема роботи реакт зображена на (Рис. 3.1.)

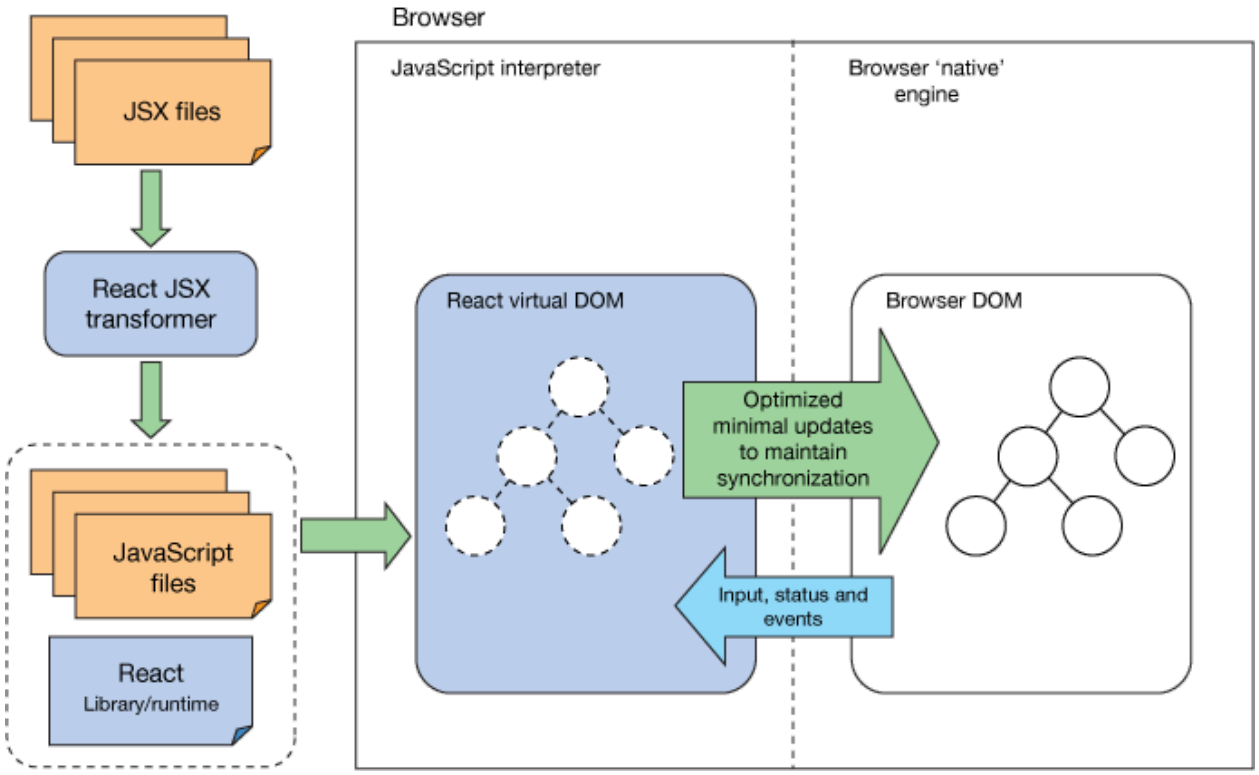

Рис. 3.1 Схема роботи React

Висока продуктивність завдяки Virtual DOM. Віртуалізуючи та зберігаючи DOM в пам'яті, React забезпечує надзвичайно швидкий рендеринг, при цьому всі зміни вигляду легко відображаються у віртуальному DOM. Спеціалізований алгоритм diff порівнює попередні та існуючі стани віртуального DOM, обчислюючи найефективніший спосіб застосування нових змін, не вимагаючи надто великої кількості оновлень. Потім вноситься мінімальна кількість оновлень для досягнення найшвидшого часу читання/запису, що призводить до загального підвищення продуктивності. На (Рис. 3.2.) Можна побачити алгоритм роботи віртуального дом

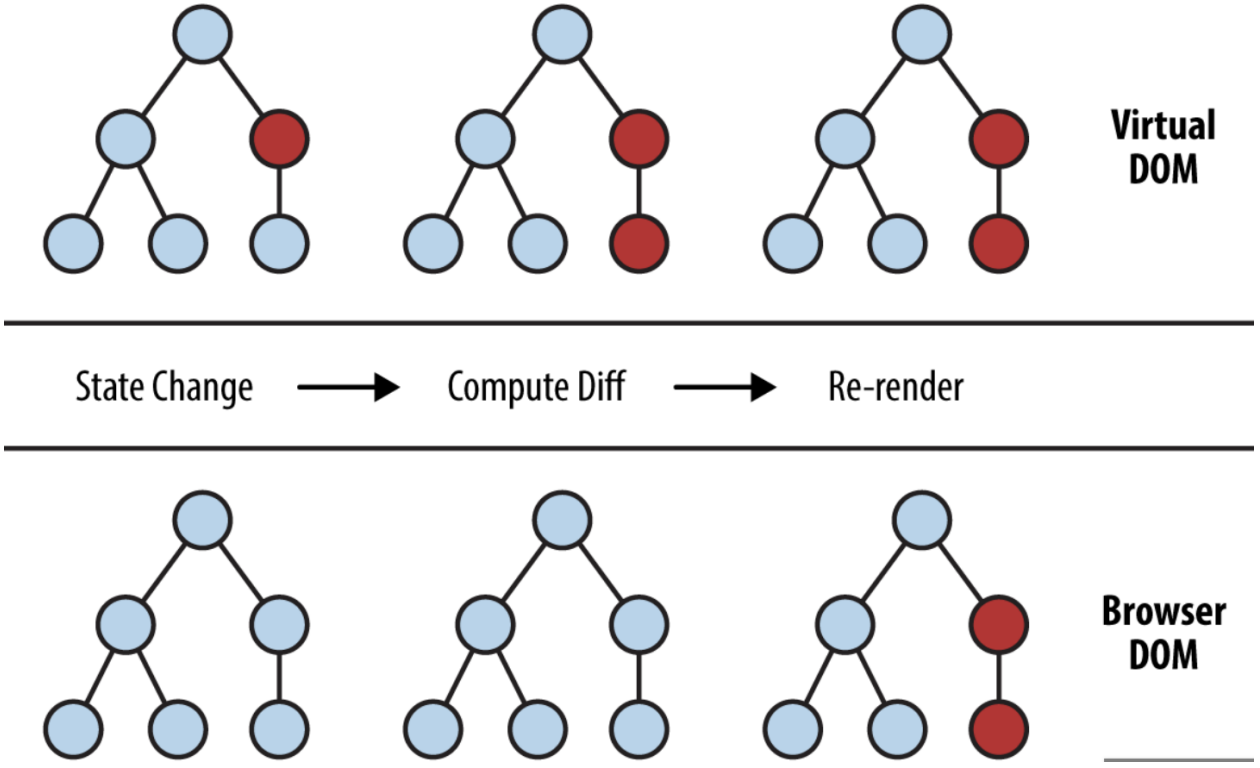

Рис. 3.2 Алгоритм порівнянь віртуального дерева

TypeScript - це надбудова над JavaScript, розроблена та відкрита компанією Microsoft у 2012 році, її основна особливість полягає в тому, що вона додає необов'язкову статичну типізацію до JavaScript, мови, яка в іншому випадку є досить вільною та поблажливою до помилок.

Microsoft зробила це, щоб допомогти вирішити проблеми зі створенням великомасштабних виробничих додатків на JavaScript, оскільки без цієї функціональності баги, помилки та проблеми легко закрадаються в код і спричиняють різні проблеми для користувачів. Використовуючи TypeScript, ми можемо зловити багато з цих проблем в коді до того, як вони потраплять у виробництво.

Окрім статичної типізації, яку TypeScript привносить в JavaScript, він також надає деякі приємні функції та інструменти, такі як краща організація коду та розширені можливості перевірки помилок, які допомагають зробити

розробку на TypeScript набагато приємнішим процесом для розробників.

Покращена якість коду та зручність супроводу

Завдяки статичній типізації, яку додає TypeScript, полегшується розуміння написаного коду. Тепер замість того, щоб дивитися на функцію і гадати, що в неї передається, можна швидко побачити, які типи функція приймає в якості аргументів, а також які типи вона повертає. Це означає, що нові розробники на проекті можуть швидко ввійти в курс справи з кодовою базою і ознайомитися з тими областями, над якими вони працюють.

Виявлення помилок

Ще одна перевага типізації полягає в тому, що вона дає вам можливість виявляти помилки і проблеми, коли вони трапляються в коді під час введення, а не чекати, поки ви запустите програму. Це означає, що ми можемо уникнути багатьох майбутніх помилок і проблем через такі дрібниці, як виклик неправильного методу для типу змінної, що дозволить нам розробляти швидше.

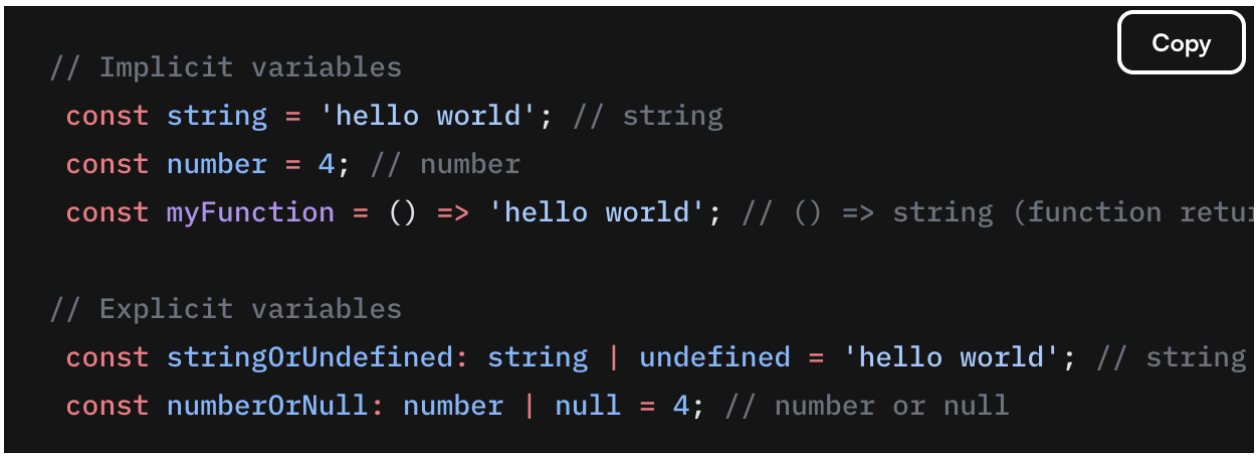

Рис. 3.3 Типи в TypeScript

Vite - це інструмент для стабілізації та оптимізації веб-пакетів. Він використовується для об'єднання всіх файлів додатку в один, що покращує продуктивність та полегшує управління кодом. Аналогами є Vite, Gulp, Bun, тощо.

Проте було обрано Vite через наступні переваги.

– Прискорює процес розробки: Якщо ми використовуємо Vite, то ваша сторінка не потребує повного перезавантаження при невеликій зміні в javascript. Таку ж перевагу можна отримати і для CSS, якщо ми будемо використовувати завантажувачі. Це також зменшило час завантаження сайту під час налагодження, і завдяки цьому можна заощадити наш час. Це усунуло проблему перезапису глобальних змінних: Ми всі знаємо, що Vite надає систему модулів, яка базується на ECMAScript (ES6). Тому кожен файл, створиний тут, стане модулем. Отже, кожна змінна, в цьому файлі, буде в локальній області видимості. Таким чином, проблема перезапису глобальних змінних, з якою ми стикалися, буде вирішена.

– Забезпечує розбиття коду: Оскільки підтримується модульна система, то файли будуть розглядатися як модулі. Оскільки файл буде розглядатися як модуль, це означає, що ми можемо використовувати можливості одного файлу в іншому. Таким чином, незважаючи на наявність різних файлів, ми можемо отримати доступ до однієї і тієї ж переваги. Таким чином, це фактично допомагає нам розділити наш код на різні модулі.

– Забезпечує мінімізацію: Мінімізація означає мінімізацію коду без зміни його функціональності. Вона видаляє всі пробіли, розриви рядків, які споживають простір. Він також видаляє непотрібний код і змінює довгі імена змінних. Таким чином, це зменшує розмір файлу і мінімізує код.

– Підтримує позначення функцій: Позначення функцій - це підхід до розробки програмного забезпечення, за допомогою якого ми відправляємо код в різні середовища під час тестування функцій. Таким чином, Vite допомагає не тільки в розробці, але і в тестуванні.

– Модулі та висока кастомізація. Завдяки великому комюніті Vite має дуже багато можливостей до кастомного налаштування

Також я

ESLint - це утиліта з відкритим вихідним кодом для лінкування Javascript, яку створив Ніколас Закас (Nicholas C. Zakas) у червні 2013 року. Вона часто використовується для пошуку проблемних патернів або коду, який не відповідає певним стильовим настановам. ESLint написана з використанням Node.js, що забезпечує швидке середовище виконання та легке встановлення за допомогою npm.

За допомогою ESLint можна запровадити стандарт кодування за допомогою певного набору окремих правил. Так, можна вмикати та вимикати ці правила. Ці правила повністю підключаються.

JavaScript, будучи динамічною мовою з вільною типізацією, особливо схильна до помилок розробників. ESLint дозволяє мені створити правила на стандарт кодування і допомагає мінімізувати синтаксичні та стилістичні помилки в коді. Основною причиною введення цих правил та стайлгайду є те, що кожен розробник має власний стиль написання (наприклад, угоди про імена/табуляцію/одинарні або подвійні лапки для рядка). І, використовуючи різні методи стилізації, ваша кодова база може виглядати дивно, бути більш схильною до помилок і вразливою. Особливо, коли працюєш з Javascript, це може призвести до проблем, з якими не хотілось би мати справу.

SCSS розшифровується як Sassy Cascading Style Sheets або Sassy CSS. Це підмножина мови CSS, яка надає більше свободи та контролю при розробці, додаючи нові можливості до CSS. Його розробив Хемптон Кетлін, а ідея належить Крісу Еппштейну та Наталі Вайценбаум.

Використовуючи SCSS вдалось розширити CSS багатьма новими можливостями, такими як змінні, вкладеність та багато інших. Завдяки всім цим додатковим можливостям, написання SCSS може в додатку реалізовано простіше і швидше, ніж це було б з написанням звичайного CSS.

Більшість браузерів не розуміють SCSS, і їх потрібно скомпілювати в CSS, перш ніж ви зможете використовувати їх у браузері. Розширення, яке використовується для таблиці стилів SCSS - .scss.

Він використовує той самий синтаксис, що і CSS, з дужками і крапкою з комою для позначення блоків і закінчень рядків. Давайте розглянемо його синтаксис на наступному прикладі:

Приклад коду на scss:

```scss
$primary: #000;
.scene {
  width: 800px;
  border-radius: 10px;
  background: #fff;
  padding: 24px;
  text-align: center;

  &-number {
     font-weight: 700;
     color: $primary;
  }
   .input {
     color: $primary;
  }

}
```

Той самий код на css

```css
.scene {
width: 800px;
border-radius: 10px;
background: #fff;
padding: 24px;
text-align: center;
```

```
}
.scene-number {
 font-weight: 700;
 color: #000;
}
.scene .input {
 color: #000;
}
```

Python - одна з найбільш універсальних, надійних та об'єктно-орієнтованих мов програмування, яка скорочує час розробки та пропонує розробникам великі можливості.

Python також називають інтелектуальною мовою програмування, оскільки вона зосереджена на двох основних речах - RAD (швидка розробка додатків) та DRY (не повторюй себе). Також, коли ви хочете склеїти або з'єднати два існуючі компоненти, можете покластися на Python.

На додаток до цього, Python також має велику підтримку спільноти Python розробників. Якщо Python розробник виявляє якісь питання або проблеми в процесі розробки, ця спільнота та онлайн-форуми приходять йому на допомогу. Крім того, бібліотеки Python, що постійно розвиваються, дозволяють розробникам створювати будь-які типи веб- та мобільних додатків, наприклад, додатки для, штучного інтелекту, машинного навчання або науки про дані.
Python легко вивчати, він добре масштабується та адаптується. Завдяки цьому вона стала однією з найбільш швидкозростаючих мов для веб- та мобільної розробки у всьому світі.

Python має просту криву навчання. Це грає на користь Python. Крім того, вона має дуже простий синтаксис.

Крім того, велика кількість онлайн-уроків, доступних для розробників, робить цю мову програмування легкою у вивченні. Крім того, це незалежна від платформи мова програмування, яку можна використовувати на великій кількості операційних систем.

Всі ці якості роблять Python легкою у вивченні та зручною у роботі мовою програмування.

Завдяки своїй послідовності та простоті вона стає ідеальним рішенням для бекенд-розробки. Широкий набір бібліотек та фреймворків економить час та зусилля розробників.

Розробники з повним стеком можуть доставляти додатки вчасно і пропонувати конкурентну перевагу своїм клієнтам.

Чудова екосистема бібліотек та фреймворків. Це найважливіша причина, чому розробники обирають Python для бекенд-розробки додатків на основі ШІ.

Бібліотека - це набір модулів із заздалегідь написаним кодом, який дозволяє розробникам використовувати його безпосередньо. Вона також дозволяє користувачам виконувати певні дії в додатку.

Python має широкий спектр готових бібліотек для використання розробниками. Розробникам не потрібно кодувати елементи базового рівня з самого початку проекту.

Додатки для штучного інтелекту та машинного навчання потребують безперервної обробки та аналізу даних.

Python має список спеціальних бібліотек, які дозволяють отримувати доступ до даних, аналізувати, обробляти та перетворювати їх. Деякі з бібліотек, які використовують розробники

Python для додатку зі ШІ

Python, незважаючи на те, що він був створений ще до появи ШІ, є найкращим вибором серед мов програмування для ШІ. Вона може похвалитися неперевершеною популярністю в галузі штучного інтелекту, особливо в

машинному навчанні - найважливішій підгалузі ШІ. Чому Python є основною мовою для розробників штучного інтелекту? Ось основні причини:

Потужний інструмент для аналізу даних: Походження Python як потужного інструменту аналізу даних робить її ідеальним вибором для компаній, що надають послуги зі штучного інтелекту. Його майстерність у роботі з великими обсягами даних зміцнила його позиції в галузі штучного інтелекту.

Фреймворки для штучного інтелекту: Чарівність Python поширюється і на наявність фреймворків для штучного інтелекту. Серед них виділяється TensorFlow, бібліотека з відкритим вихідним кодом, створена виключно для розробки машинного навчання. Вона чудово підходить для навчання та розгортання глибоких нейронних мереж.

Інші фреймворки, орієнтовані на ШІ, включають

– scikit-learn: Універсальний інструмент для навчання моделей машинного навчання.

– PyTorch: Найкраще підходить для обробки візуальних даних та природної мови.

– Keras: Кодовий інтерфейс, що спрощує складні математичні обчислення.

– Theano: Бібліотека, призначена для визначення, оптимізації та оцінки математичних виразів.

Простота вивчення та використання: Python славиться своєю зручністю у використанні. Це одна з найпростіших мов для вивчення і роботи з нею, що робить її чудовим вибором для досвідчених розробників і новачків у сфері ШІ.

У світі програмування ШІ, що постійно розвивається, Python залишається вірним супутником, надаючи розробникам можливість створювати передові рішення для ШІ та сприяючи успіху послуг з розробки ШІ. Логотип мови python можна побачити на (Рис. 3.4)

Рис. 3.4. Логотип Python

3.2. Підготовка Середовища

Підготовка середовища розробки є важливим етапом при розробці. Ось кілька кроків та інструментів, які були використані для створення сприятливого середовища розробки:

Розширення для VSCode

Для підвищення продуктивності та зручності розробки були встановлені різні розширення, такі як ESLint для перевірки коду, Prettier для автоматичного форматування коду, розширення для підтримки React та SCSS.

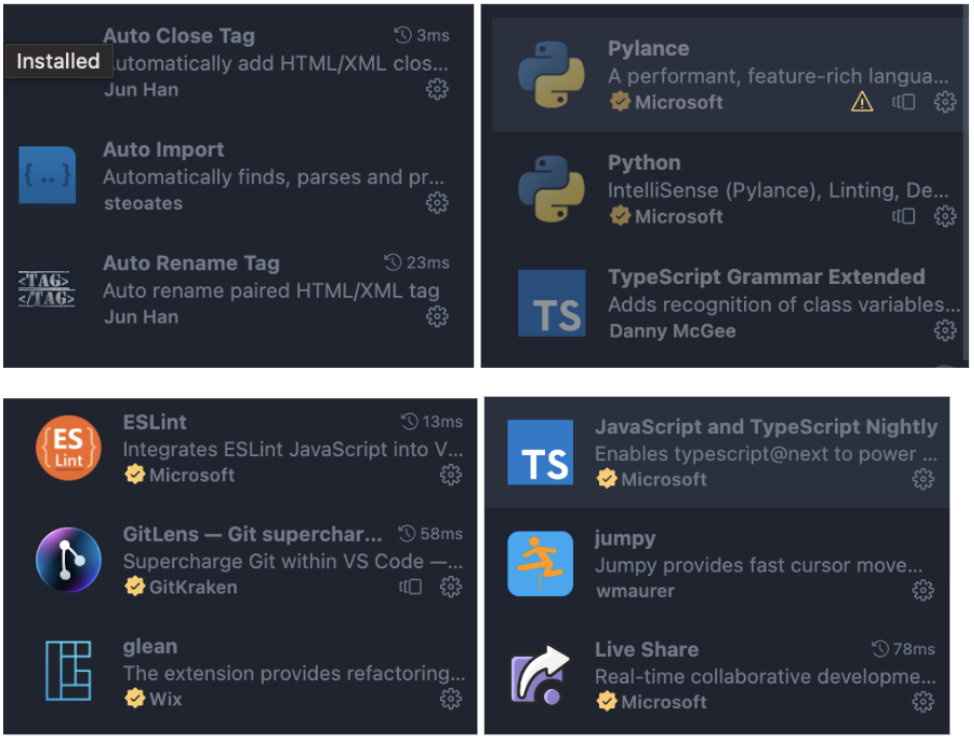

Рис. 3.5. Список розширень IDE

Node.js - це фреймворк, який дозволяє розробляти веб-додатки на JavaScript. Він використовує однопоточний, одноядерний модель виконання, що робить його ідеальним для розробки асинхронних додатків, які працюють в реальному часі.

Node.js має ряд переваг, які роблять його популярним вибором для розробників веб-додатків. Ось деякі з цих переваг:

– швидкість. Node.js є дуже швидким фреймворком, який може обробляти великі обсяги даних.

– ефективність. Node.js є дуже ефективним фреймворком, який використовує мінімум ресурсів.

– платформа. Node.js є кросс-платформеним фреймворком, який можна використовувати для розробки додатків для різних платформ.

Node.js можна використовувати для розробки різних типів веб-додатків, включаючи:

– веб-сайти. Node.js можна використовувати для розробки статичних і динамічних веб-сайтів.

– веб-сервіси. Node.js можна використовувати для розробки веб-сервісів, які надають доступ до даних або функцій.

– мобільні додатки. Node.js можна використовувати для розробки мобільних додатків, які працюють на пристроях iOS і Android.

Ось деякі конкретні приклади того, як Node.js може бути використаний при розробці веб-додатків:

– для створення асинхронних додатків, які працюють в реальному часі. Наприклад, Node.js можна використовувати для розробки чат-ботів, які можуть обробляти повідомлення в реальному часі.

– Для створення веб-додатків, які використовують великі обсяги даних. Наприклад, Node.js можна використовувати для розробки додатків для обробки даних, таких як системи логістики або управління клієнтами.

– для створення веб-додатків, які працюють на різних платформах. Наприклад, Node.js можна використовувати для розробки додатків для хмарних платформ, таких як AWS або Azure.

У цілому, Node.js є потужним і універсальним фреймворком, який може використовуватися для розробки різних типів веб-додатків.

PNPM (Package Manager for Node.js) - це менеджер пакетів для Node.js, який є альтернативою популярному менеджеру пакетів NPM. PNPM має ряд переваг перед NPM, включаючи:

– швидкість. PNPM в середньому на 20-30% швидший за NPM.

– ефективність. PNPM використовує менше пам'яті і ресурсів, ніж NPM.

– простота використання. PNPM має простіший інтерфейс, ніж NPM.

PNPM можна використовувати для установки, видалення та управління пакетами Node.js. Він також можна використовувати для управління конфігурацією ваших проектів Node.js.

Ось деякі конкретні приклади того, як PNPM може бути використаний при розробці веб-додатків:

Для швидкої установки та оновлення пакетів. PNPM можна використовувати для швидкої установки та оновлення пакетів, що може заощадити час і ресурси.

Для управління конфігурацією проектів. PNPM можна використовувати для управління конфігурацією ваших проектів Node.js, що може зробити ваші проекти більш універсальними і масштабованими.

У цілому, PNPM є потужним і ефективним менеджером пакетів для Node.js, який може використовуватися для розробки веб-додатків.

Ось деякі додаткові переваги використання PNPM:

1. PNPM підтримує версіонування пакетів за допомогою префіксів. Це дозволяє вам встановлювати і використовувати кілька версій одного пакету в одному проекті.
2. PNPM має вбудований менеджер конфігурації. Це дозволяє вам зберігати конфігурацію вашого проекту в єдиному файлі.
3. PNPM має вбудований менеджер пакетів для npm. Це дозволяє вам використовувати PNPM для управління пакетами, які доступні в npm.

Якщо ви шукаєте менеджер пакетів для Node.js, який є швидким, ефективним і простим у використанні, PNPM є хорошим вибором.

Git - це система керування версіями, яка дозволяє відстежувати зміни в коді. GitLab є популярним інструментом для зберігання та спільної роботи над кодом. Це означає, що він має велику спільноту користувачів, що забезпечує доступ до підтримки та ресурсів.

Він українській. Підтримуйте Українське. Слава Україні!

Загалом, використання Git та GitLab є хорошим вибором для розробки будь-якого масштабного веб-додатку. Ці інструменти пропонують широкий

спектр функцій, які можуть допомогти вам поліпшити продуктивність і якість вашого додатку.

Visual Studio Code (VSCode)

Він є легким та потужним текстовим редактором, який підтримує багато розширень та плагінів, що полегшують розробку.

VS Code - це крос-платформний редактор коду, який може легко працювати на macOS, Windows та Linux. Всі веб-технології VS Codes в кінцевому підсумку використовують Electron Framework, що означає, що додатки, створені за допомогою VS Code, є гнучкими і безпроблемними при їх оновленні.

VS Code порівняно набагато швидший за Visual Studio. Він може завантажуватися швидше, якщо порівнювати з IDE.

Visual Studio Code неймовірно гнучкий. Він може робити практично все, що захоче розробник. Незважаючи на те, що вона розроблена як редактор коду, можна повторити те, що робить інтегроване середовище розробки.

VS Code, порівняно з Visual Studio, є досить простим і зручним з точки зору досвіду розробки. VS Code досить впорядкований і зрозумілий, тому розробник не заплутається в будь-яких складнощах.

Visual Studio Code - найкращий вибір для веб-розробки. Він пропонує неймовірну підтримку з тисячами інструментів та розширень, які готові працювати на вашу користь.

Для деяких розробників більшість функцій Visual Studio Code є несуттєвими, незважаючи на те, наскільки чудовим є VS Code. Для них Visual Studio є найкращим вибором.

Visual Studio функціонально багата. У більшості випадків Visual Code ідеально відповідає вимогам більшості розробників без необхідності покладатися на додаткові розширення або плагіни.

У Visual Studio дуже легко співпрацювати з усією командою, як під час розробки, так і під час налагодження коду. Робочий процес є надзвичайно

плавним і наповнений всіма видами функцій, які можуть знадобитися в довгостроковій перспективі.

Коли справа доходить до аналізу, налагодження та профілювання продуктивності складного коду, Visual Studio є неймовірним варіантом.
Visual Studio досить часто використовується в індустрії розробки ігор. Наприклад, UNITY, багатоплатформенне середовище, інтегроване з Visual Studio, дозволяє без особливих зусиль створювати крос-платформні мобільні ігрові додатки, AR/VR додатки та багато іншого!

Робоче середовище: Було обрано плагіни та налаштування відповідно до потреб та вимог проекту, створив робочі папки для організації коду та
встановив зручні скрипти для розробки, тестування та розгортання.

Підготовка середовища розробки включала в себе встановлення та налаштування інструментів, що дозволили розробнику зосередитися на написанні високоякісного коду та швидкому розвитку додатку "Ideations".

3.3. Atomic Design Methodology

Під час розробки фронтенду додатку "Ideations" структура папок була розроблена з використанням методології Atomic Design. Цей підхід є найкращим на Q3 2023, тому цей підхід було спрощено під проект

Що таке атомарний дизайн? Методологія атомарного дизайну, створена Бредом Фростом, - це методологія проектування для створення надійних систем дизайну з чітким порядком та ієрархією. Як випливає з назви, вона походить від базової хімічної концепції - складу всієї матерії.
Atoms (Атоми). У Цій категорії знаходяться найбільш базові компоненти, які не можна поділити на менші елементи. Це, наприклад, кнопки, текстові поля, іконки та інші елементи, які використовуються як будівельні блоки для інших компонентів.

Molecules (Молекули). У цій категорії зберігаються компоненти, які включають в себе кілька атомів та об'єднують їх разом для вирішення конкретних завдань. Наприклад, поле для введення тексту разом з кнопкою пошуку.

Organisms (Організми). У Організмах відбувається більше складний збір атомів та молекул, створюючи цілісні компоненти, які можуть бути використані на сторінках додатку. Наприклад, навігаційне меню зі списком пунктів.

Примітка. Було уникнуто частини молекули і організми в данній архітектурі, тому що вони є надлишковими, іноді це складно відрізнити де що використовувати

Templates (Шаблони). Шаблони в системах дизайну допомагають визначити стандартний макет сторінки для декількох сторінок зі схожою функціональністю. Простіше кажучи, шаблони - це стандартизовані макети для організації атомів, молекул і організмів у продукті. Витративши час на стандартизацію макетів сторінок, мені вдалося значно покращити узгодженість продукту та зменшути надлишковість коду. Якщо подивимтись на сторінку товару на будь-якому сайті електронної комерції або на профіль будь-якого виконавця на Spotify, то побачимо, що всі вони мають схожу структуру. Це чудово, тому що користувачам не потрібно щоразу знайомитися з новими макетами, а розробники можуть динамічно генерувати вміст сторінки без необхідності повторного переписування коду. Таким чином в Ideations ми можемо перевикористовувати шаблони

Pages (Сторінки). Мені подобається думати про шаблони як про скелетну структуру, а про сторінки - як про плоть зверху. Сторінки в атомарному дизайні - це екземпляри шаблонів у вашому користувацькому інтерфейсі, але з високою точністю. Сторінки - це те, що ваші користувачі побачать у готовому продукті. Сторінки приймають кілька шаблонних форм і допомагають дизайнерам думати про різні стани, яких можуть набути наші базові атоми і молекули. Дозвольте мені проілюструвати, як це відбувається на прикладі Spotify.

Використання Atomic Design Methodology сприяє покращенню організації та перевикористовуваності компонентів, забезпечує легке розширення та зберігання структури інтерфейсу. Такий підхід робить розробку більш ефективною та системною. Приклад компонентів можна побачити на (Рис. 3.6)

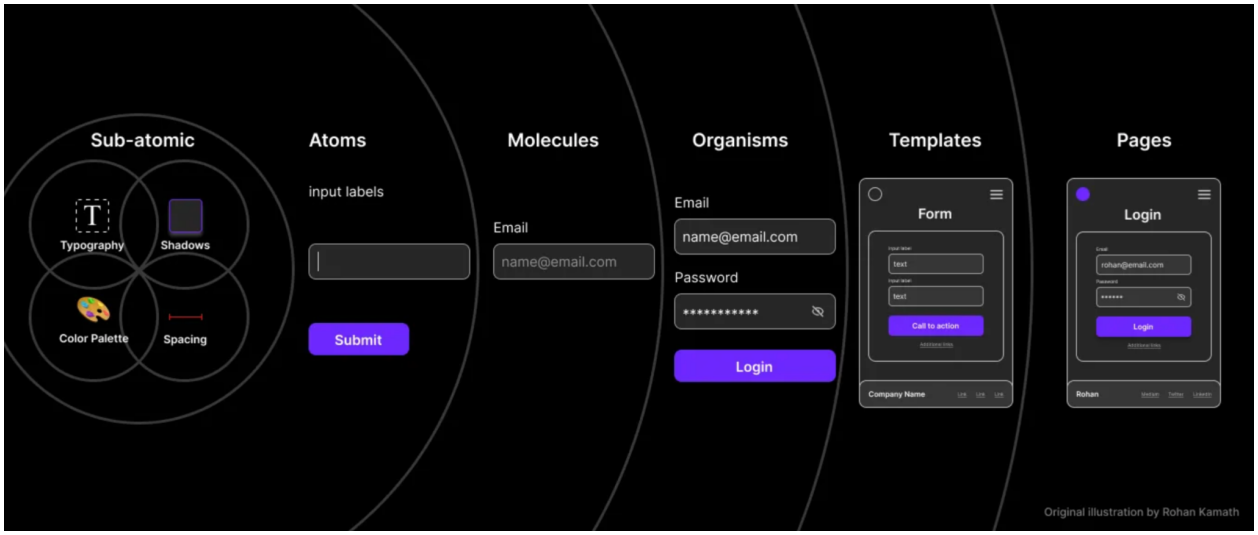

Рис. 3.6. Atomic Design Methodology

У реалізованому проекті була застосована структура папок, що відображає принципи Atomic Design. Atomic Design - це методологія, розроблена Бредом Фростом, яка допомагає розробникам організовувати компоненти інтерфейсу за рівнями складності, від найпростіших (атомів) до складних (шаблонів та сторінок).

Структура Папок:

1. Atoms: Найпростіші компоненти, такі як кнопки, іконки, поля вводу. Ці елементи використовуються як будівельні блоки для створення більш складних компонентів.

2. Molecules: Комбінації атомів, що формують молекули. Наприклад, форма пошуку, що складається з поля вводу та кнопки.

3. Organisms: Ще більш складні компоненти, які об'єднують молекули та атоми. Наприклад, заголовок веб-сторінки, що включає логотип, навігацію (молекула) та кнопки (атоми).

4. Templates: Структурні шаблони, які визначають розташування організмів та молекул на сторінці.

5. Pages: Кінцеві сторінки, що використовують шаблони з реальним змістом та дані. Це те, що користувач бачить на кінцевому продукті.

Важливість Структури Папок:

1. Організація та Підтримуваність: Чітка структура папок сприяє організації коду, роблячи його більш читабельним та легким для підтримки.

2. Повторне Використання Компонентів: Атомарний підхід до дизайну забезпечує високий рівень повторного використання компонентів, знижуючи потребу в дублюванні коду.

3. Масштабованість: Завдяки модульній природі, проект легко масштабується, дозволяючи додавати нові компоненти або модифікувати існуючі без значного переписування коду.

4. Консистентність: Ця структура сприяє візуальній та функціональній консистентності веб-додатку, що є ключовим для створення зручного користувацького інтерфейсу.

5. Ефективність Розробки: Методологія передбачає розбиття інтерфейсу на окремі компоненти, які можна розглядати як "атоми", з яких побудовані більш складні "молекули", "організми", та інші елементи інтерфейсу. Модульність, яку надає Atomic Design, дозволяє розробникам легко керувати та підтримувати код. Кожен компонент має визначену функціональність і може використовуватися в різних частинах додатку без необхідності переписування коду. Це полегшує розвиток та підтримку додатку, а також дозволяє команді розробників працювати паралельно над різними компонентами.

Організована структура папок, відповідно до принципів Atomic Design, сприяє зручній співпраці в команді. Кожна група компонентів (атоми, молекули, організми тощо) має свою власну папку, що спрощує пошук та редагування коду. Розробники можуть концентруватися на роботі з конкретними частинами

додатку, не втрачаючи час на навігацію в масиві файлів. Структуру папок для організації компонентів зображно на рис. 3.7.

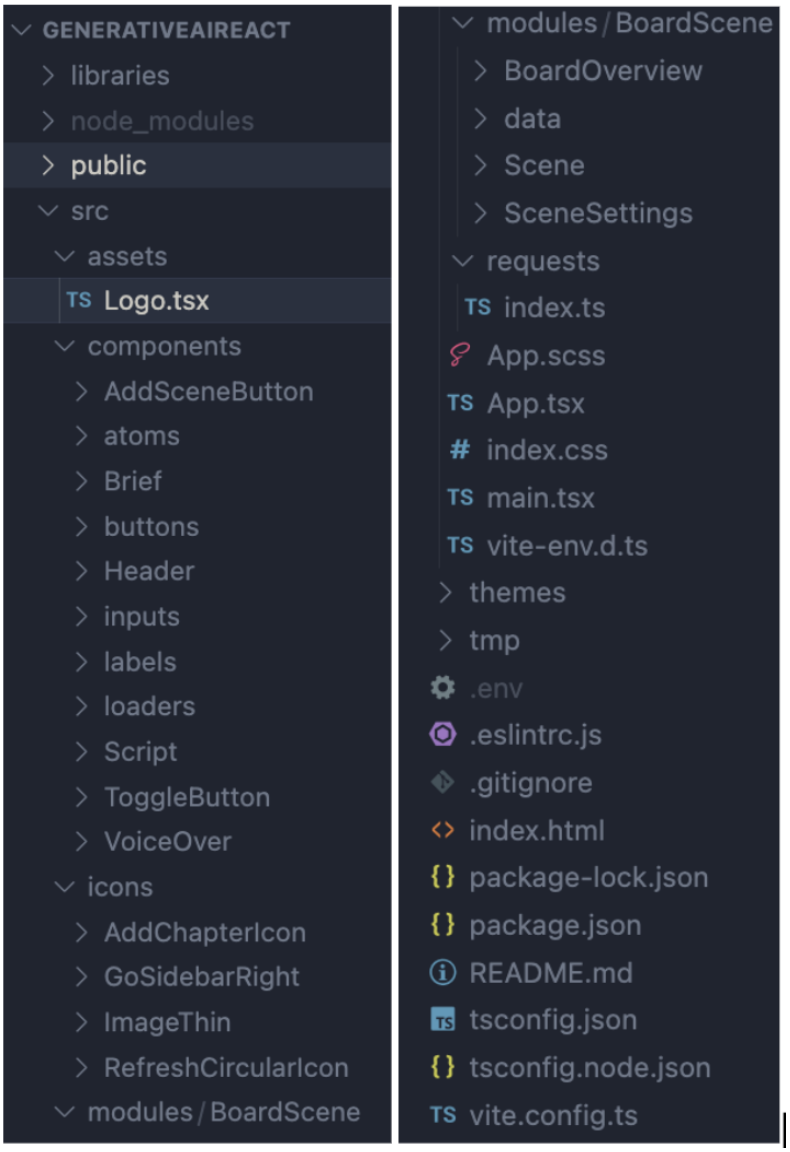

Рис. 3.7. Файлова структура проекту

3.4. Фронтенд розробка

Компонент Header в додатку "Ideations" є частиною інтерфейсу, яка відображає верхню частину веб-сайту чи додатку. Його основна функція - це відображення логотипу компанії чи бренду, який ідентифікує додаток "Ideations". Основні характеристики цього компонента такі:

Імпорт Логотипу: Компонент імпортує логотип з файлу "@assets/Logo". Це означає, що логотип був передварительно створений та збережений як ресурс в проекті.

CSS Стилі: Для стилізації компонента Header використовуються CSS стилі, які були описані в окремому файлі "styles.scss". Це дозволяє задавати вигляд та розміщення логотипу та інших елементів в шапці додатку.

Структура Компонента: Компонент Header складається з основного контейнера <header>, в якому розташовується контейнер для логотипу . Логотип сам представлений компонентом <Logo />, який імпортується з ресурсів.

Логотип: Основна функція цього компонента - відображення логотипу в шапці додатку. Він може бути оформлений згідно брендового стилю компанії "Ideations".

```
import { Logo } from "@assets/Logo";
import './styles.scss'

export const Header = () => {
  return (
    <header className='header-container'>
      <div className="logo-container">
        <Logo />
      </div>
    </header>
  );
};
```

"BoardOverview," відповідає за відображення та управління інтерфейсом для створення та редагування сценарію та розкадровки в режимі "Storyboard."

Основні функціональність та особливості цього компонента включають:

1. Сценарій та Розкадровка: Компонент дозволяє користувачеві редагувати сценарій (script) та розкадровку (scenesArr). Сценарій містить опис сцен, а розкадровка представляє собою набір сцен, які візуалізують описи з сценарію.

2. Додавання та Редагування Сцен: Користувач може додавати нові сцени за допомогою кнопки "Add scene" і редагувати описи сцен.

3. Динамічне Оновлення Даних: Компонент автоматично оновлюється, коли змінюється довжина сценарію, кількість сцен або інші дані, що можуть вплинути на відображення розкадровки.

4. Завантаження Зображень: Є можливість завантаження зображень для сцен за допомогою функції "generateImages." Ця функція дозволяє відобразити візуальний контент для кожної сцени.

5. Інтерактивне Управління: Компонент надає можливість змінювати описи сцен та додавати нові сцени між існуючими.

6. Завантаження Сценарію: Компонент може завантажувати існуючий сценарій та відображати його на інтерфейсі для подальшого редагування.

7. Спеціальний Інтерфейс для Завантаження: Якщо сценарій ще не завантажений або завантаження в процесі, компонент відображає анімаційний інтерфейс ("SceneSceleton"), що ілюструє, що дані ще завантажуються.

```
export const Scene = ({..}) => {
…
 return (
   <>
    <div className={containerClassName}
      ref={sceneRef}
    >
      <div className="scene-number">
```

```
        Scene {index + 1}
      </div>
      <div className="image-container">
        {activeImage?.img64 ? (
          <>
            <img src={`${activeImage?.img64}`}
              alt="image"
              className='image-preview' />
          </>
        ) : (
              <> {isFetchingImages ? <Skeleton width={360} height={360} /> : <div
classNam="image-preview placeholder"><ImageThin className='icon'
/></div>}</>
        )}
        <section className="images-thumbnails" >
          {renderThumbnails()}
            <div className="image-thumbnail-container">
              <Skeleton width={83} height={83} />
            </div>
        </section>
      </div>
      <div className="input-container">
        {isFetchingScript ? <Skeleton width={750} height={55} /> : (
          <>
            <TextAreaWithLabel
              labelProps={{ value: 'Description' }}
              inputProps={{
                disabled: true,
                value: sceneData.description,
```

```
            onChange: ({ target }) => changeSceneDescription(index, target.value)
          }} />
        <IconButton
          onClick={regenerateImage}
          disabled={isFetchingImages}
          icon={<RefreshCircularIcon className='icon' />} />
      </>
    )}
  </div>
  <div className="voice-over-input-container">
    {isFetchingScript ? <Skeleton width={750} height={55} /> : (
      <TextAreaWithLabel
        labelProps={{ value: 'Voice Over' }}
        inputProps={{
          disabled: true,
          value: sceneData.voiceOver,
          onChange: ({ target }) => changeSceneDescription(index, target.value)
        }} />
    )}
  </div>
</div>
{index + 1 < scenes.length && (
  <AddSceneButton
    onClick={() => addScene(index + 1)}
    className='add-scene-between-btn'
    icon={<AddChapterIcon className="icon" />}
  >Add scene
  </AddSceneButton>
)}
```

```
  </>
);
};
```

Компонент `SceneSceleton` є важливим елементом дипломної роботи, оскільки він виконує роль заглушки або плейсхолдера, що використовується під час завантаження відповідного контенту. Використання скелетних екранів є сучасною практикою у веб-дизайні, яка покращує користувацький досвід, мінімізуючи дискомфорт від часу очікування завантаження контенту.

В контексті дипломної роботи, ось що робить `SceneSceleton` цінним:

1. Покращення UX/UI: Він вносить вагомий внесок у користувацький інтерфейс (UI) та користувацький досвід (UX), забезпечуючи плавність інтерфейсу, коли реальний контент ще не завантажено.
2. Зниження когнітивного навантаження: Скелетні екрани допомагають знизити когнітивне навантаження на користувачів, надаючи їм візуальні підказки про те, що контент ще завантажується, що знижує імпульсивність і відчуття чекання.
3. Застосування передових технологій: Компонент демонструє розуміння та застосування передових технологій в дизайні, що є важливим для сучасного веб-розробника.
4. Візуальна консистентність: Забезпечує візуальну консистентність у додатку, показуючи, як розміщення реального контенту виглядатиме після завантаження, що підтримує загальний дизайн інтерфейсу.
5. Код та стилізація: Подання компоненту з використанням SCSS підкреслює використання модульного та масштабованого підходу до стилізації, що є показником професіоналізму і дозволяє легко адаптувати стилі для різних розмірів екрану та пристроїв.

6. Технічна реалізація: Впровадження такого компоненту є прикладом технічної компетентності розробника у створенні реактивних інтерфейсів, які відгукуються на стан завантаження додатку.

7. Підвищення продуктивності: Компонент сприяє підвищенню продуктивності роботи користувачів з додатком, оскільки вони отримують зворотний зв'язок відразу і знають, що додаток працює над відповіддю на їх запит.

Включення `SceneSceleton` в дипломну роботу показує, що автор має практичні навички у створенні сучасних веб-інтерфейсів, вміє реалізовувати користувацькі вимоги та вирішувати реальні дизайнерські проблеми.

```
import classnames from 'classnames';
import { Skeleton } from '#components/loaders/Skeleton';
import './styles.scss'

export const SceneSceleton = () => {
  const containerClassName = classnames('scene-sceleton scene');

  return (
    <div className={containerClassName}>
      <div className="scene-number">
        <Skeleton width={80} height={27} />
      </div>
      <div className="image-container">
        <Skeleton width={350} height={350} />
      </div>
      <div className="input-container">
        <div className="label">
          <Skeleton width={100} height={17} />
```

```
          </div>
          <Skeleton width={750} height={50} />
        </div>
      </div >
  );
};
```

```
{
  "name": "generativeaireact",
  "private": true,
  "version": "0.0.0",
  Debug
  "scripts": {      You, 3 months ago • Save changes after 2 days of development ;-)
    "dev": "vite",
    "build": "tsc && vite build",
    "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@html-eslint/eslint-plugin": "^0.19.1",
    "classnames": "^2.3.2",
    "eslint-config-airbnb": "^19.0.4",
    "eslint-plugin-html": "^7.1.0",
    "lodash.debounce": "^4.0.8",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loading-skeleton": "^3.3.1",
    "react-pro-sidebar": "^1.1.0-alpha.1",
    "react-query": "^3.39.3",
    "uuid": "^9.0.1",
    "zustand": "^4.4.1"
  },
  "devDependencies": {
    "@types/lodash.debounce": "^4.0.7",
    "@types/react": "^18.2.15",
    "@types/react-dom": "^18.2.7",
    "@typescript-eslint/eslint-plugin": "^6.0.0",
    "@typescript-eslint/parser": "^6.0.0",
    "@vitejs/plugin-react": "^4.0.3",
    "eslint": "^8.45.0",
    "eslint-import-resolver-typescript": "^3.6.0",
    "eslint-plugin-import": "^2.28.1",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.3",
    "tsconfig-paths": "^4.2.0",
    "typescript": "^5.0.2",
    "vite": "^4.4.5"
  }
}
```

Рис. 3.8. Бібліотеки проекту

Цей код представляє набір змінних SCSS, які використовуються для створення тематизації та забезпечення послідовності стилів у веб-додатку. SCSS (Sassy CSS) — це препроцесор, який додає потужність та елегантність до звичайного CSS, дозволяючи використовувати змінні, вкладення, міксини тощо для більш ефективного керування стилями.

Змінні, визначені тут, включають колірну палітру додатку, що містить первинні, вторинні та третинні кольори, а також кольори для тексту, акцентів та системних повідомлень (наприклад, помилок та успіху). Окрім того, встановлені змінні для визначення відступів, що допомагає підтримувати консистентність та простоту у масштабуванні інтерфейсу. Наприклад:

– `$primary` і відповідні кольори (`$primary-text`, `$primary-hover` тощо) визначають основні кольори бренду для використання по всьому додатку.

– `$error` і `$success` визначають кольори, що використовуються для повідомлень про помилки та успішні дії відповідно.

– `$padding-*` змінні визначають стандартизовані розміри відступів для елементів, що дозволяє легко змінювати відступи без необхідності переписування багатьох стилів.

– `$button-bg` та `$button-color` визначають фоновий колір та колір тексту кнопок, асоціюючи їх із первинною колірною схемою.

Використання цих змінних у SCSS дозволяє розробникам легко адаптувати та масштабувати стилі по всьому додатку, забезпечуючи консистентність та полегшуючи процес оновлення дизайну.

```
$primary: #1C3F70;
$primary-text: #1C3F70;
$primary-accent: #f0f0f0;
$primary-hover: #2d5aa3;
```

```
$primary-active: #172c4e;
$primary-box-shadow: 0px 0px 4px 4px rgba(0, 0, 0, 0.1);

$secondary: #81beff;
$secondary-accent: #000;
$secondary-hover: #a6d8f2;
$secondary-active: #6bbad7;

$tertiary: #F2F2F2;

$tertiary: #F2F2F2;
$accent: #ff741d;
$error: #FF4C4C;
$success: #2ECC71;
$text: #333;

$padding-1: 4px;
$padding-2: 8px;
$padding-3: 16px;
$padding-4: 24px;
$padding-5: 32px;

$button-bg: $primary;
$button-color: $primary-accent;
```

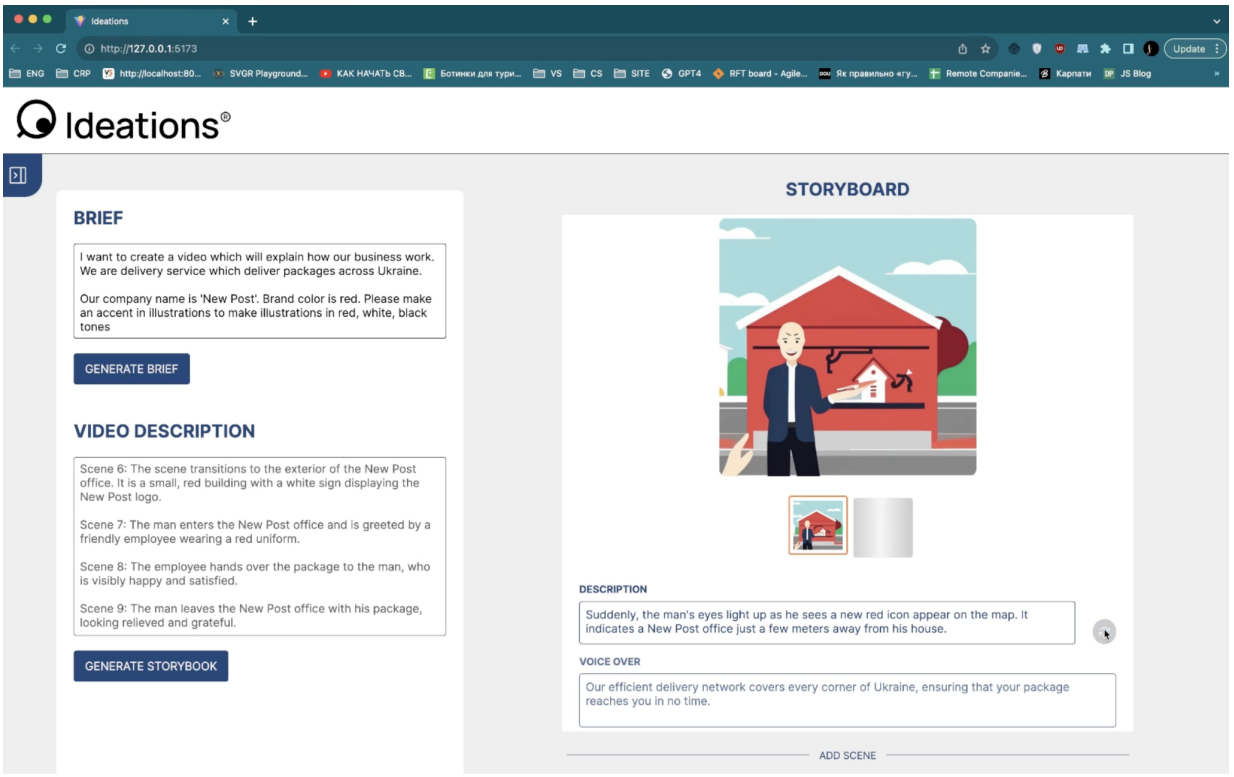

Рис. 3.9. Користувацький інтерфейс додатку

3.5. Бекенд розробка

Модуль **Generator**. Це Сервер веб-додатку, який може створювати зображення на основі текстових підказок. Існує клас `Generator`, який є базовим для генерації зображень, та два класи, які його розширюють: `DummyGenerator` для тестування функціоналу та `StableDiffusionGenerator`, який використовує модель штучного інтелекту "Stable Diffusion" для створення зображень, керованих текстом. Користувачі можуть надіслати текстовий опис (prompt), і сервер за допомогою моделі AI створить відповідне зображення, яке повертається у форматі base64 через API, створене за допомогою фреймворку FastAPI.

```
import random
import base64
import io
```

```
import uvicorn
from PIL import Image
from fastapi import FastAPI

try:
  import torch
  from torch.nn import DataParallel
  from diffusers import DiffusionPipeline
except ImportError:
  print('Diffusion models are not available')


class Generator:
  """
  Клас Generator - це абстрактний клас, який містить метод __call__.
  """
  def __call__(self, prompt: str, example_img: Image.Image = None) -> Image:
    raise NotImplementedError


class DummyGenerator(Generator):
  """
    Клас DummyGenerator - це клас-нащадок класу Generator, який просто повертає зображення з диску.
  Використовуєтсья для тестування.
  """
  def __call__(self, prompt, example_img=None) -> Image:
    if example_img is None:
      image = Image.open('astronaut_on_horse.png')
```

```
        return image
    else:
        width, height = example_img.size
                    random_color = (random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255), int(255 * 0.3))
        overlay = Image.new('RGBA', (width, height), random_color)
        result = Image.alpha_composite(example_img.convert('RGBA'), overlay)
    return result


class StableDiffusionGenerator(Generator):
  """
    Клас StableDiffusionGenerator - це клас-нащадок класу Generator, який
використовує модель Stable Diffusion для генерації зображень.
   При ініціалізації класу завантажується модель з хабу, яка зберігається в полі
pipe.

   При виклику класу викликається метод __call__, який приймає prompt - текст,
який буде використовуватись для генерації зображення.
  """
  def __init__(self):
    self.pipe = DiffusionPipeline.from_pretrained(
        "CompVis/stable-diffusion-v1-4",
        torch_dtype=torch.float16,
        use_safetensors=True,
    ).to("cuda:0")

  def __call__(self, prompt, example_img=None) -> Image:
    res = self.pipe(prompt).images[0]
```

```
    return res


if __name__ == '__main__':
    """
    Для комунікації з генератором використовується FastAPI та реалізовано один ендпоінт /generate, який приймає POST-запити
    і викликає метод __call__ класу Generator і повертає зображення у форматі base64.
    """
    app = FastAPI()
    generator = StableDiffusionGenerator()


    @app.post("/generate/")
    def generate(prompt: str = "", example_img=None):
        pil_img = generator(prompt, example_img)
        buffered = io.BytesIO()
        pil_img.save(buffered, format="PNG")
        image_base64 = base64.b64encode(buffered.getvalue()).decode("utf-8")

        return {"img64": image_base64}


    uvicorn.run(app, host="0.0.0.0", port=8080)
```

Модуль Communicator. Виступає інтерфейсом для взаємодії зі StableDiffusion для генерації зображень. Він ініціалізується хостом і портом, на

якому запущено сервіс. Метод `generate` надсилає POST-запит із текстовим запитом (і, за бажанням, зображенням) до сервісу. Сервіс, ймовірно, модель штучного інтелекту, генерує зображення на основі запиту і повертає його у форматі base64. Генератор-комунікатор декодує це зображення і перетворює його в об'єкт PIL-зображення для подальшого використання або відображення в додатку. Таке налаштування дозволяє динамічно генерувати зображення, які можна використовувати для різноманітних додатків, наприклад, для створення візуального контенту з текстових описів.

```
import base64
import io

import requests
from PIL import Image



class GeneratorCommunicator:
    """
    Клас GeneratorCommunicator - це клас, який використовується для взаємодії з генератором. Він надає зручний інтерфейс.

    При ініціалізації класу задається порт, на якому запущений генератор.

     Метод generate приймає prompt - текст, який буде використовуватись для генерації зображенняу
    та, можливо, example_img - зображення, яке теж буде використовуватись для генерації зображення.

    Метод повертає згенероване зображення, отримане від генератора.
```

```
    """
    def __init__(self, port, host='localhost'):
        self.port = port
        self.host = host

    def generate(self, prompt, example_img=None):
        req = requests.post(f'http://{self.host}:{self.port}/generate',
                            json={'prompt': prompt, 'example_img': example_img})
        if req.status_code != 200:
            raise Exception(f'Generator error: {req.text}')
        base64_image_string = req.json()['img64']
        image_bytes = base64.b64decode(base64_image_string)
        image_buffer = io.BytesIO(image_bytes)
        image = Image.open(image_buffer)
        return image


if __name__ == '__main__':
    communicator = GeneratorCommunicator(8080)
    res = communicator.generate('test')
    print(res)
```

Модуль Перетворює текстові описи на зображення та скрипти за допомогою ШІ. У ньому є компонент, який відповідає за зберігання та пошук зображень, а також компонент, який генерує зображення з підказок, використовуючи комунікаційний міст до генератора, ймовірно, моделі ШІ. Він використовує API OpenAI для створення відеоскриптів з наданих користувачем брифів. Додаток забезпечує безпеку перехресних запитів і обробку помилок.

Кінцевим результатом є зображення або сценарій, який повертається користувачеві у певному форматі, інтегруючи можливості ШІ з користувацьким вкладом для створення персоналізованого контенту.

Промпт для ChatGPT: “Instructions: Act as a professional scriptwriter who will write the scripts for a video storyboard based on the client brief.
The videos will be 2d vector art. Do not use any proper nouns in scene description. Use only generic descriptions which describes the image.
Do not include anything except the script. No notes, no comments, nothing. Give explicit instructions for the illustrations which will fit perfectly as a prompt for stable diffusion.
Do not describe scenes based on previous scenes. Each scene must be described independently. The script should be written in the present tense.
Each scene must be one point in time. The scene cannot be described as any process or a series of events, it should be one point in time.

Example of lines of the output: “1: a man stands on the roof. it is dark. he wears a suit. he holds a gun. he looks at the city. he is sad.
2: full moon. birds fly visually though it
So each like starts with a number and a colon. Then there is a description of the scene.
Don't use word "scene" in descriptions. Just discard the image in details and don't focus on it being a future video.
Always be specific. Don't give options to choose from.”

3.6. Оптимізація моделі

Для того щоб покращити моделі і створювати кращі ескізи. Було прийнято рішення оптимізувати модель, надавши їй данні для оптимізації з 2d анімованих відео. Цей процес називається fine tuning.

Fine tuning у контексті машинного навчання та штучного інтелекту - це

процес налаштування та оптимізації попередньо навченої моделі під конкретні завдання або датасети. Воно включає коригування параметрів моделі, яка вже була навчена на великому загальному наборі даних, для досягнення кращих результатів на специфічних даних або завданнях.

Під час fine tuning, модель "оптимізується" шляхом додаткового тренування на новому, часто більш обмеженому наборі даних. Це дозволяє моделі краще адаптуватися до особливостей цих нових даних, зберігаючи при цьому загальне знання, набуте під час первісного тренування.

Цей процес особливо корисний у ситуаціях, коли доступ до великої кількості маркованих даних є обмеженим або коли необхідно адаптувати модель до специфічних потреб. Fine tuning є популярною практикою у галузях, які займаються обробкою природної мови, комп'ютерним зором, розпізнаванням мовлення та іншими областями, де моделі глибокого навчання можуть виявити складні закономірності у даних.

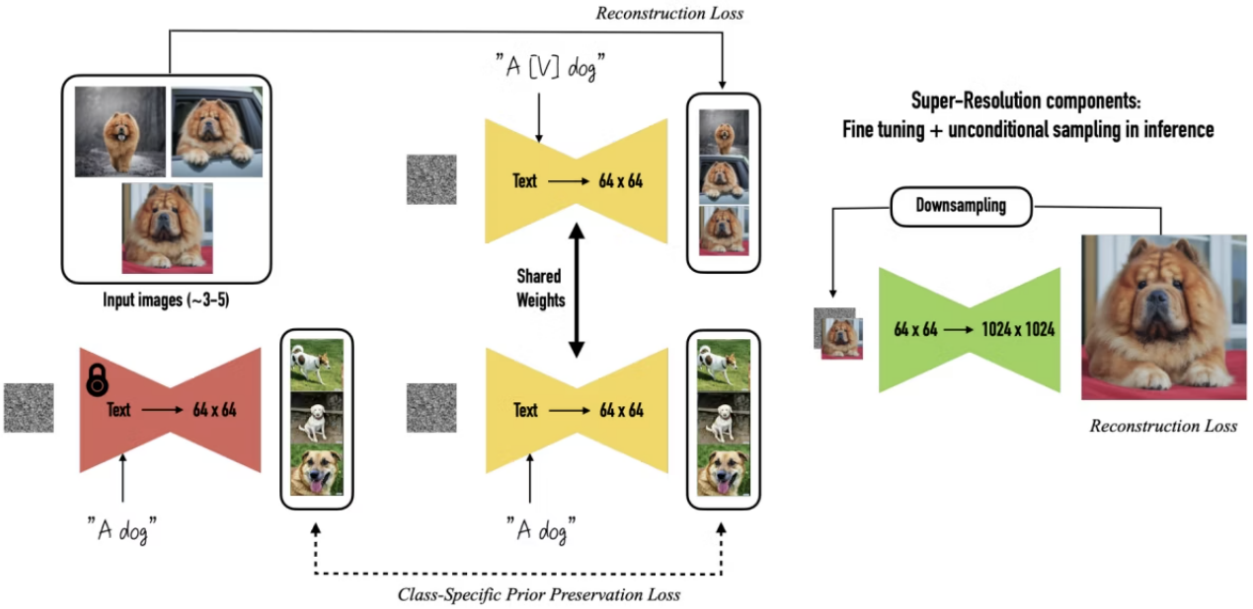

Рис. 3.10. Приклад оптимізації моделі для генерації зображення

Збір даних для оптимізації та тренування моделей генеративного штучного інтелекту, зокрема для створення 2D анімованих відео, є важливим етапом в розробці подібних додатків. Ось детальний опис способів та процесу збору таких даних:

Створення Датасету: Перший крок - це створення датасету, що складатиметься з великої кількості вхідних даних і відповідних вихідних даних. У вашому випадку, це можуть бути вхідні описи сцен для анімації та відповідні зображення 2D сцен, що були створені на основі цих описів. Існують наступні способи збору данних.

1. Ручний Збір Даних: Ручний збір даних є традиційним методом у створенні анімаційних сцен, де художники та аніматори вручну розробляють кожну сцену, відповідно до детальних описів. Цей підхід, хоча й вимагає значних витрат часу та коштів, забезпечує високий рівень творчого контролю та унікальності кожної анімаційної сцени. Він дозволяє детально працювати над кожним аспектом візуалізації, від колориту до елементів дизайну, що особливо цінно при створенні високоякісного та індивідуального контенту..

2. Генерація Даних За Допомогою AI: генерація даних за допомогою AI відкриває нові можливості в створенні анімаційних сцен. Використання існуючих моделей генеративного штучного інтелекту, як у вашій моделі "Ideations", дозволяє автоматизувати процес створення анімацій на основі текстових описів. Цей метод значно зменшує час і витрати на виробництво, дозволяючи швидко генерувати візуальний контент. Хоча генеративний AI може бути обмежений у плані деталізації та індивідуального стилю порівняно з ручним методом, він пропонує значну гнучкість та ефективність, особливо при масштабуванні проектів або при швидкому прототипуванні ідей.

3. Збір Даних З Інтернету: Використання даних з Інтернету включає пошук і збір вже існуючих відео або анімаційних сцен, які можуть бути використані як навчальні матеріали для моделі. Цей метод може виявитися корисним для отримання великої кількості даних за короткий час. Він дозволяє

дослідникам швидко збільшити обсяг даних, доступних для тренування моделі, що може бути особливо корисним у випадках, коли збір унікальних даних виявляється складним або дорогим. Однак, існує ризик, що зібрані дані можуть бути не цілком відповідними або можуть мати варіативну якість, що може вплинути на ефективність навчання моделі.

4. Ручна Анотація Даних: Цей процес включає додаткову ручну обробку даних, де художники або анотатори вносять корективи або позначки в дані, щоб забезпечити їх відповідність вимогам проекту. Ручна анотація необхідна для підвищення якості та точності даних, особливо в контексті тренування моделей машинного навчання, де точність даних має критичне значення. Цей процес може бути трудомістким, але він забезпечує більш точний контроль над якістю навчальних датасетів, що може в кінцевому результаті значно підвищити ефективність навченої моделі. Ручна анотація дозволяє виправляти неточності та неоднозначності, які можуть виникати при автоматичному зборі даних, забезпечуючи, що модель тренується на високоякісних та релевантних даних.

5. Доступ до Джерел Зображень: Іноді можна отримати доступ до джерел зображень або відео з дозволом для використання їх для навчання вашої моделі. У такому випадку, важливо дотримуватися авторських прав та ліцензій.

6. Збір Метаданих: Крім самого зображення або анімаційної сцени, важливо збирати метадані, такі як описи сцен, категорії, ключові слова тощо. Це допоможе вам краще керувати та організовувати ваш датасет.

7. Обробка та Перевірка Даних: Зібрані дані повинні бути оброблені, очищені від шуму, відформатовані та перевірені на наявність помилок.

8. Резервні Копії та Зберігання Даних: Забезпечення безпеки даних та створення резервних копій є важливою частиною процесу збору даних.

Це лише загальний огляд процесу збору даних для fine-tuning моделей генеративного штучного інтелекту, і важливо ретельно розглянути кожен конкретний випадок для вирішення специфічних завдань та потреб вашої

дипломної роботи.

Щоб навчити модель SD генерувати 2D-зображення, спочатку потрібно зібрати набір даних 2D-зображень. Було обрано збирати данні з інтернету. Для цього було створено скрипт, який робить скріншот відео кожні 10 секунд, а потім обробляє ці скріншоти в сегменти розміром 200x200 пікселів. Ця колекція зображень слугуватиме навчальною вибіркою для моделі.

Було прийнято рішення використовувати спосіб номер 3. Для цього був створений скрипт для парсингу зображення з 2d анімованих відео

```
def capture_screenshots(video_path, interval=10):
    """ Capture screenshots from the video at given intervals. """
    cap = cv2.VideoCapture(video_path)
    fps = cap.get(cv2.CAP_PROP_FPS)
    success, image = True, None

    while success:
        success, image = cap.read()
        if not success:
            break
        frame_id = int(round(cap.get(cv2.CAP_PROP_POS_FRAMES)))
        if frame_id % (fps * interval) == 0:
            yield image

def segment_image(image, segment_size=(200, 200)):
    """ Segment an image into smaller parts. """
    img_height, img_width, _ = image.shape
    for y in range(0, img_height, segment_size[1]):
        for x in range(0, img_width, segment_size[0]):
            yield image[y:y + segment_size[1], x:x +
segment_size[0]]

def process_video(video_path, output_path, interval=10,
segment_size=(200, 200)):
    """ Process a video file to capture and segment images. """
```

```
    if not os.path.exists(output_path):
        os.makedirs(output_path)

    for idx, screenshot in
enumerate(capture_screenshots(video_path, interval)):
        for jdx, segment in enumerate(segment_image(screenshot,
segment_size)):
            segment_path = os.path.join(output_path,
f'segment_{idx}_{jdx}.png')
            cv2.imwrite(segment_path, segment)
```

Розглянемо приклад збору данних в цій роботі використовуючи відео. Вибране для цього відео завантажується в файлову систему, де воно стає доступним для обробки за допомогою спеціалізованого скрипта. Цей скрипт, створений з використанням сторонніх бібліотек, автоматизує процес вилучення зображень з відео. Конкретно, кожні 10 секунд він вибирає одне зображення з відеопотоку.

Особливістю цього процесу є те, що скрипт не просто вилучає повні кадри, а обробляє їх, вирізаючи фрагменти розміром 200x200 пікселів. Цей розмір було обрано з міркувань оптимальності для подальшої обробки та аналізу. Приклад зображення донора демонструється на рис. 3.11.

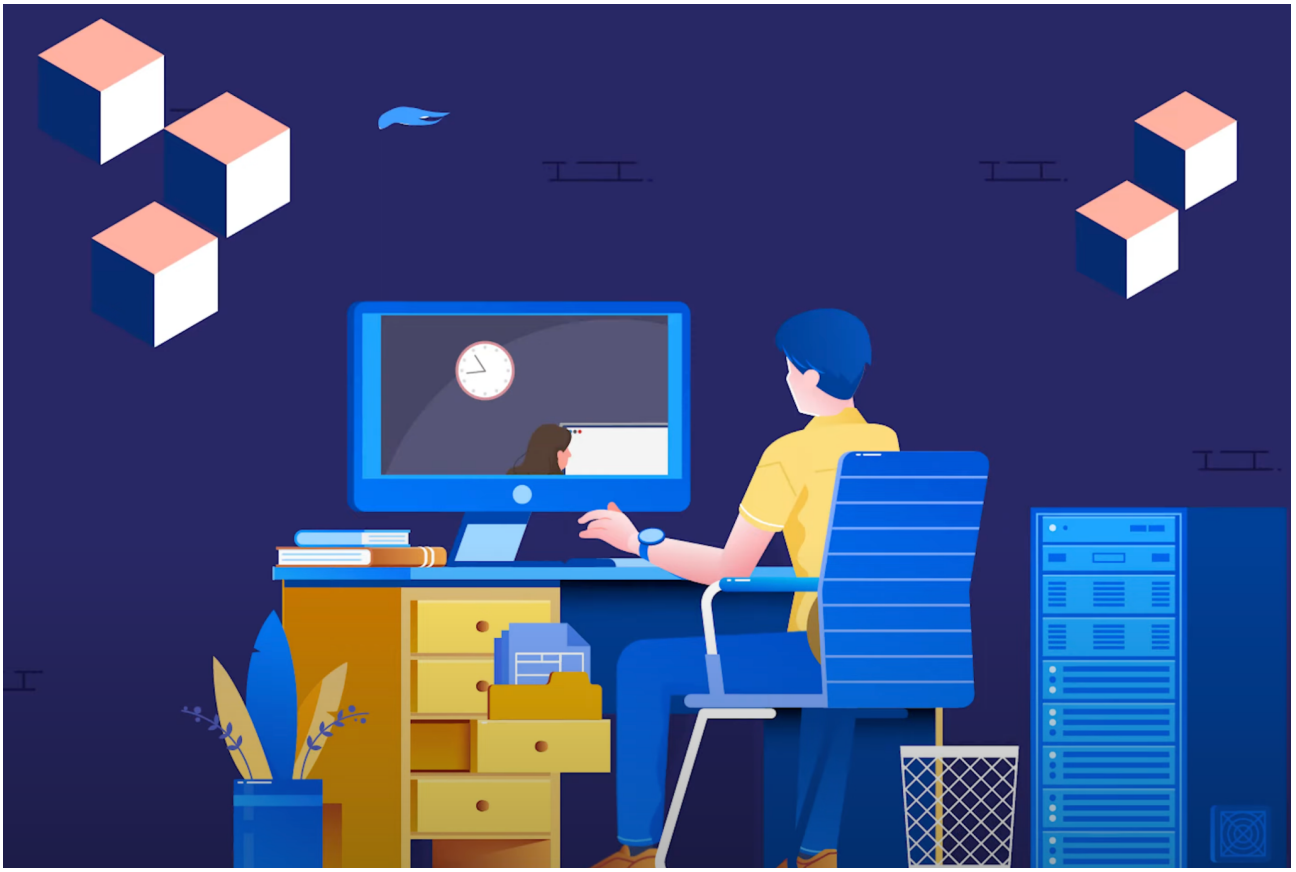

Рис. 3.11. Приклад зображення з відео

Після завершення процесу вилучення, отриманий масив нарізаних зображень готовий до завантаження в інструмент для fine tuning моделі stable diffusion. Fine tuning - це крок, який дозволяє адаптувати модель під конкретні задачі або типи даних, підвищуючи точність та якість генерацій, які вона виробляє. У даному випадку, за допомогою цього процесу, модель навчається генерувати більш точні та релевантні зображення на основі специфічних характеристик датасету, отриманого з відео. Цей підхід дозволяє значно покращити результати, які може забезпечити генеративний штучний інтелект, роблячи його більш ефективним для конкретних задач анімації або візуалізації.

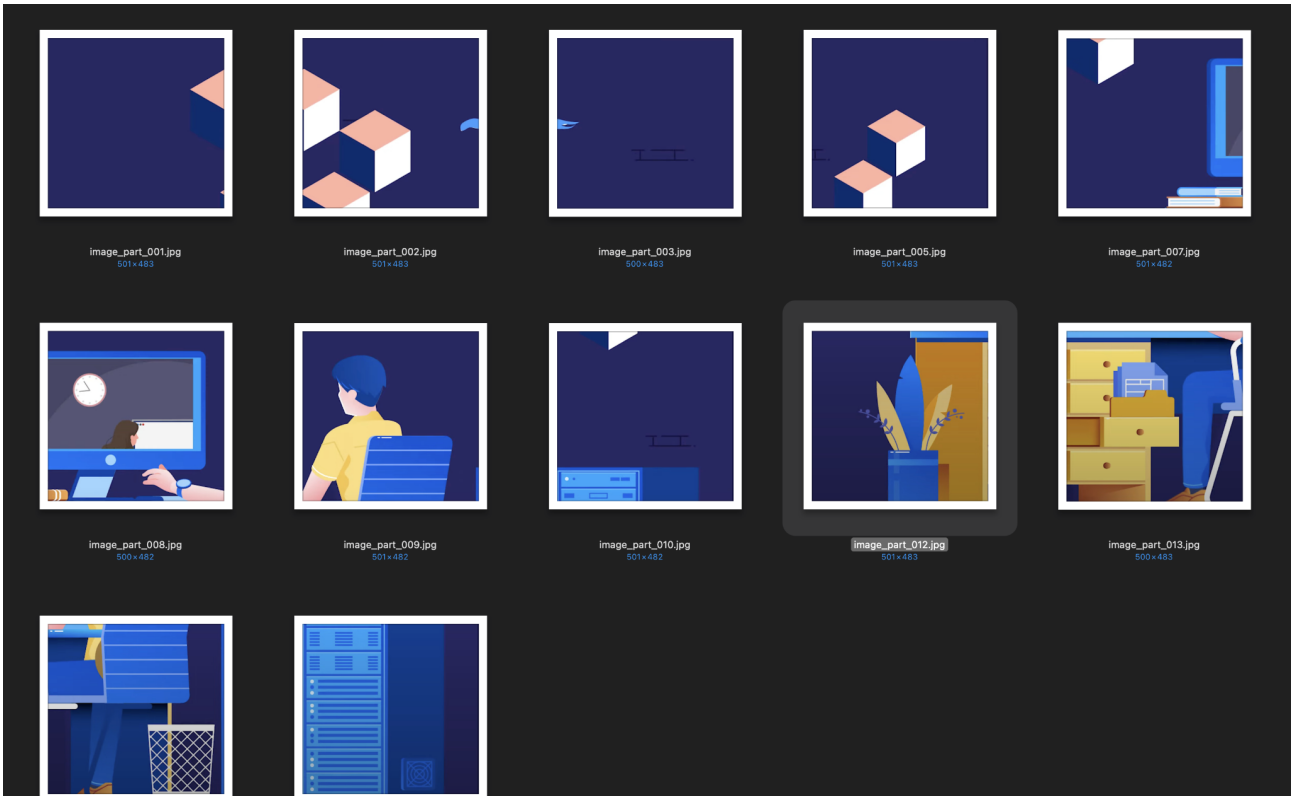

Рис. 3.12. Приклад масиву нарізанних зображень

Далі ви використовуєте ці зображення для точного налаштування моделі StableDiffusion, який включає в себе регулювання ваг моделі, щоб вона вивчила специфічні характеристики ваших 2D-зображень. Процес налаштування є ітеративним і вимагає декількох раундів навчання, під час яких модель генерує зображення, а потім вносяться корективи на основі того, наскільки ці зображення відповідають бажаному 2D-стилю.

В майбутньому планується значно покращити процес збору даних шляхом вдосконалення фільтрації датасету. Основна мета цього покращення полягає у видаленні неінформативних зображень, що не несуть значущого внеску у тренування моделі. Це може включати зображення з поганою якістю, нерелевантні кадри, або такі, що містять шум або інші візуальні спотворення.

Підвищення якості датасету через фільтрацію є важливим кроком для підвищення точності та надійності генеративних моделей. Видалення неінформативних зображень допоможе зменшити обсяг даних, які модель має

обробляти, тим самим зробивши процес навчання більш ефективним і скорочуючи час, необхідний для тренування. Крім того, вибіркове виключення таких зображень сприятиме зменшенню шуму в датасеті та підвищенню загальної якості генерованих зображень.

Плани щодо покращення збору даних в майбутньому також включають розробку алгоритмів та методів, які автоматично визначатимуть та відсіюватимуть неінформативні зображення. Це може включати використання технік машинного навчання для визначення важливості та релевантності кожного зображення в контексті конкретних вимог до тренування моделі. Ці інновації не лише підвищать якість датасету, але й забезпечать більш гнучку та автоматизовану систему збору даних, що є важливим для масштабування проекту та реалізації його потенціалу у майбутньому.

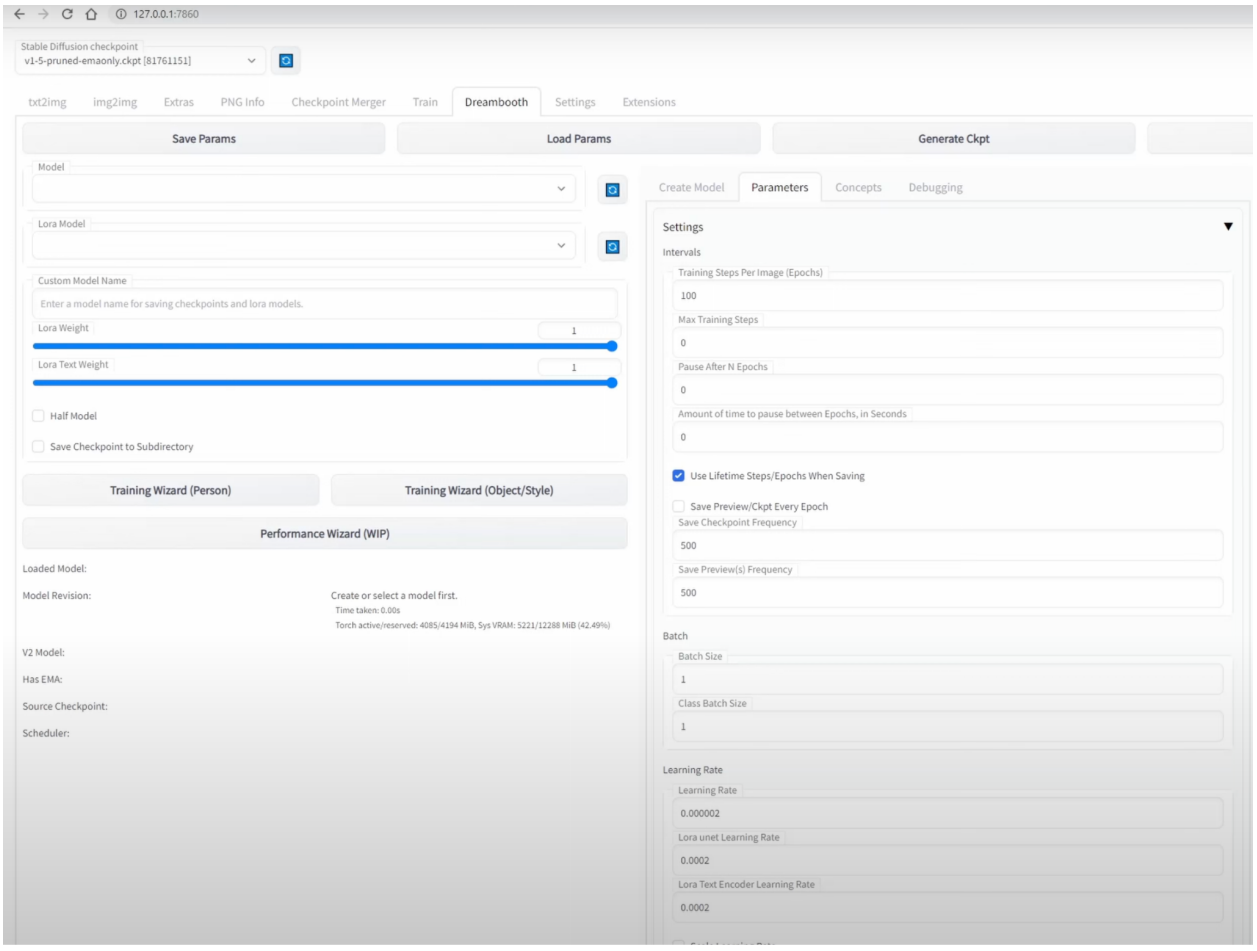

Рис. 3.13. Користувацький інтерфейс для оптимізації моделі

Додаток взаємодіє з моделлю, щоб генерувати і рендерити зображення, які відповідають наданим користувачем описам, переводячи текст в промпт і генеруючи стилізовані 2D-зображення.

Фінальна частина включає інтеграцію цієї навченої моделі у внутрішню частину програми, де вона може отримувати дані підказок з інтерфейсу користувача, генерувати зображення на основі підказок, використовуючи вивчений 2D-стиль, а потім повертати це зображення у фронтенд для відображення користувачеві.

ВИСНОВОК ДО РОЗДІЛУ 3

У розділі 3 було зроблено вибір технологій, таких як React, TypeScript для фронтенду та Python для бекенду, був критично важливим для забезпечення ефективності та функціональності додатку. Підготовка середовища в Visual Studio Code з використанням розширень та інструментів, як-от Vite та ESLint, підкреслювала важливість гнучкості та високої якості коду.

Застосування методології Atomic Design у поєднанні з адаптованою імплементацією дозволило створити інтуїтивно зрозумілий та модульний інтерфейс користувача. Фронтенд розробка зосереджувалася на використанні React та TypeScript для створення динамічного та адаптивного UI, в той час як бекенд, реалізований на Python, а ефективність обробки даних.

Ключовим аспектом було тренування моделі Stable Diffusion, що дозволило додатку "Ideations" ефективно генерувати зображення відповідно до текстових підказок користувачів. Ця функціональність стала визначальною для забезпечення інноваційності та практичності додатку у сфері створення візуального контенту.

Таким чином, уважне планування, аналіз та виконання кожного етапу розробки "Ideations" сприяли створенню веб-додатку, який відкриває нові можливості в створенні відео та сприяє креативності людей, без необхідності витрачати багато багато годин на вивчення спеціальних інструментів та практи.

ВИСНОВКИ

Було проведено всебічне дослідження ГШІ та розробка веб-додатку на основі ГШІ під назвою "Ideations", спрямованого на використання генеративного штучного інтелекту для створення 2D-зображень. Аналіз трьох ключових розділів дозволяє сформулювати комплексне уявлення про проект та його значення.

У першому розділі було викладено теоретичну основу ГШІ, де глибокий аналіз історії, алгоритмів та моделей ШІ виявив їх потенціал та вплив на сучасні технології. Осмислення цих аспектів забезпечило міцний фундамент для вибору та використання відповідних технологій у додатку.

У другому розділі було сформовано потреби користувачів, де в результаті детального аналізу був сформульований функціонал додатку. Особлива увага була приділена користувацькому інтерфейсу, дизайну та впровадженню передових технологій для оптимізації користувацького досвіду.

У третьому розділі було підшотовлено середовище розробки та вибір сучасних технологій, які були використані для фронтенду та бекенду. Тренування моделі було ключовим моментом для забезпечення інноваційної здатності додатку генерувати високоякісні зображення.

Проект втілює в собі синтез теоретичних та практичних знань, що веде до створення додатку, який не лише задовольняє потреби користувачів, але й відкриває нові можливості в області цифрового дизайну та креативності. Ця робота становить важливий внесок у сферу штучного інтелекту та його практичного застосування, а також надає перспективу для подальшого розвитку та інновацій у цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Книга: Захарія О.В. Основи машинного навчання та штучного інтелекту: навч. посіб. Київ: Академія, 2020.
2. Книга: Flanagan, David. "JavaScript: The Definitive Guide." O'Reilly Media, 2020
3. Книга: Alpaydin, Ethem. "Introduction to Machine Learning." 2020.
4. Книга: Лутц М. "Навчаємося програмувати на Python." К.: 2019.
5. Книга: Гудфеллоу І., Бенджіо І., Курвіль А. "Глибоке навчання." Харків: Видавництво «ІнТЕХ», 2019. 800 с.
6. Книга: Васильєв А. "Програмування на Python для початківців." К.: Діалектика.
7. Ресурс Інтернету: Спільнота українськомовних розробників Python. URL: https://python-ukr.org/.
8. Ресурс Інтернету: A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT https://arxiv.org/pdf/2303.04226.pdf
9. Ресурс Інтернету: Stable Diffusion Documentatio. URL: https://stablediffusionapi.com/docs
10. Ресурс Інтернету: Блог з корисними відео про веб-дизайн та UX/UI. URL: https://www.youtube.com/@WAYUPIN/
11. Ресурс Інтернету: Канал з відео про програмування. URL: https://www.youtube.com/@AboutProgramming
12. Ресурс Інтернету: Chat GPT API Documentation. URL: https://platform.openai.com/docs
13. Ресурс Інтернету: ESLint: Pluggable JavaScript linter." URL: https://eslint.org/docs/rules/

14. Стаття: Гудман Б., Хакев Б. "Застосування генеративного штучного інтелекту в комп'ютерному мистецтві." Журнал "Комп'ютерна графіка і мультимедіа", 2020

15. Стаття: Сідер П.М., Стоун А., Бек Г. "Генеративні моделі в машинному навчанні." 2019

16. Стаття: Марія Іванова. "Застосування генеративних моделей у візуальному мистецтві." Журнал "Мистецтво та Технології", 2019.

17. Навчальний посібник: Чоллет Ф., Курс Ж. "Поглиблене навчання з PyTorch." Київ: Видавництво "Діалектика", 2020. 400 с.

18. Ресурс Інтернету: Articel GAI. URL: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

19. Ресурс Інтернету: React - A JavaScript library for building user interfaces." URL: https://reactjs.org/

20. Ресурс Інтернету: Документація Python. URL: https://docs.python.org/3/

21. Ресурс Інтернету: Machine Learning Papers. URL: https://paperswithcode.com/